PERANCANGAN KONTROL GERAKAN TABUNG TOMOGRAF MENGGUNAKAN MIKROKONTROLER AT89C51 DENGAN METODE LOGIKA *FUZZY*

TUGAS AKHIR

OLEH :

Moh. Samsun Afifuddin

NRP : 2294.100.084

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

2000

PERANCANGAN KONTROL GERAKAN TABUNG TOMOGRAF MENGGUNAKAN MIKROKONTROLER AT89C51 DENGAN METODE LOGIKA *FUZZY*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik Elektro
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
S u r a b a y a

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Ir. MURDI ASMORO ADJI	Ir. TASRIPAN, MT.
NIP. 130 532 014	NIP. 131 918 685

S U R A B A Y A

PEBRUARI, 2000

ABSTRAK

Tomograf adalah peralatan dalam dunia kedokteran untuk mengambil foto *rontgen* untuk mendapatkan gambar diagnostik suatu lapisan khusus dari jaringan/obyek yang tertutup oleh lapisan/obyek lainnya. Ini diselesaikan dengan memanfaatkan peralatan tambahan yang memungkinkan tabung sinar-x dan film bergerak dengan berporos pada titik *fulcrum* selama *ekspose*. Hasil radiografnya disebut tomogram, yang memperlihatkan sebuah gambar jelas obyek yang terletak di dalam bidang khusus dan mengaburkan obyek yang berada di atas dan di bawah bidang khusus tersebut.

Gerakan tabung yang membutuhkan waktu lama bisa menyebabkan perubahan posisi pasien/obyek selama ekspose, yang menyebabkan posisi fokus berubah, pada akhirnya hasil gambar yang diperoleh tidak jelas. Juga paparan radiasi sinar-x yang lama dapat menimbulkan bahaya kesehatan. Peralatan tomograf yang ada saat ini membutuhkan waktu relatif lama dalam pengambilan gambar, rata-rata 3 detik untuk pergerakan tabung 60^0.

Dalam tugas akhir ini telah dibuat kontrol *fuzzy* untuk mengatur gerakan tabung sinar-x pada peralatan tomograf sehingga didapatkan waktu *ekspose* yang lebih cepat. Sistem yang dibuat terdiri atas mikrokontroler AT89C51 yang diprogram untuk berfungsi sebagai kontrol *fuzzy* sekaligus kontrol sistem secara keseluruhan, dan pembangkit sinyal PWM untuk kontrol kecepatan motor arus searah, ADC, *driver* arah dan kecepatan motor, dan sensor posisi sebagai umpan balik.

Dari perancangan ini dihasilkan sebuah kontroler *fuzzy* dengan maksimum empat masukan dan dua keluaran berbasis mikrokontroler AT89C51 yang murah dan menunjukkan kerja yang cukup baik. Kecepatan *software fuzzy* 2,5 ms per siklus dengan 4 masukan, 2 keluaran, dan 5 *rule fuzzy*. Waktu gerakan yang terukur dalam peralatan yang telah dibuat adalah 1,2 detik pada gerakan 30^0 dan 1,68 detik pada gerakan 60^0, dan kesalahan pengaturan posisi yang terjadi rata-rata 2 %.

Kontrol *fuzzy* memudahkan perancangan sistem kontrol karena meniru cara berpikir manusia, daripada menggunakan model matematis. Penggunaan mikrokontroler sebagai kontrol *fuzzy* perlu ditingkatkan, karena dapat mereduksi biaya dibandingkan bila menggunakan kontrol *fuzzy* jadi. Disarankan penggunaan sensor posisi yang lebih teliti untuk mengurangi kesalahan posisi yang terjadi.

KATA PENGANTAR

Puji syukur senantiasa penyusun panjatkan kepada Allah SWT, karena atas limpahan rahmat dan hidayah-Nya sehingga dapat menyelesaikan tugas akhir yang berjudul **"PERANCANGAN KONTROL GERAKAN TABUNG TOMOGRAF MENGGUNAKAN MIKROKONTROLER AT89C51 DENGAN METODE LOGIKA *FUZZY*"** sebagai prasyarat untuk meraih gelar sarjana pada Jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember Surabaya.

Dalam menyusun Tugas Akhir ini penulis senantiasa mendapat bantuan dan dukungan yang sangat berharga. Untuk itu penulis mengucapkan banyak terima kasih, terutama kepada :

- Ibu, Bapak, serta adik-adikku yang telah memberikan dukungan semangat serta doa sehingga tugas akhir ini dapat terselesaikan.
- Bapak Ir. Murdi Asmoro Adji selaku Dosen Pembimbing I yang telah bersedia mengorbankan waktunya dan dengan sabar memberikan bimbingan.
- Bapak Ir. Tasripan, MT. selaku Pembimbing II yang senantiasa memberikan bimbingan dan dorongan semangat kepada penulis.
- Bapak Ir. Nawantowibowo selaku dosen wali.
- Bapak Ir. Soetikno, selaku Koordinator Bidang Studi Elektronika Jurusan Teknik Elektro ITS.
- Bapak Dr.Ir. Achmad Yazidie, M.Eng. selaku Ketua Jurusan Teknik Elektro FTI-ITS.
- Rekan-rekan mahasiswa, khususnya di Bidang Studi Elektronika.
- Bapak-bapak di bagian Elektromedik Instalasi Perawatan Sarana dan bagian Radiologi RSUD Dr. Soetomo Surabaya yang telah banyak membantu dalam penyelesaian tugas akhir ini.
- Bapak Dr.Ir. Daniel M. Rosyid dan Ibu Ratna serta rekan-rekan di Apotik Sepuluh Nopember dan Poliklinik ITS yang telah banyak memberikan bantuan dan dukungan.
- Rekan-rekan di Masjid Manarul Ilmi ITS, serta semua pihak yang tak bisa penulis sebutkan satu persatu, yang telah memberikan bantuan dan dukungan sehingga tugas akhir ini dapat terselesaikan.

Masih banyak kekurangan di dalam tugas akhir ini yang membutuhkan pengembangan dan perbaikan, untuk itu penyusun menerima dengan terbuka segala kritik dan masukan demi kesempurnaannya.

Akhirnya penulis berharap semoga Tugas Akhir ini memberikan manfaat bagi semua, khususnya untuk perkembangan dunia ilmu.

Surabaya, Februari 2000

Penyusun

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dunia kedokteran tidak lepas dari bidang radiologi. Radiologi digunakan dalam diagnosa untuk mengambil gambar bagian dalam tubuh pasien, seperti paru-paru, ginjal, jantung kerusakan tulang ataupun bagian dalam kepala. Pada kasus tertentu, dibutuhkan gambar organ tubuh pada suatu lapisan tertentu dengan mengabaikan lapisan lain di bawah dan di atasnya, untuk keperluan ini digunakan alat tomograf.

Pengaturan gerakan tabung tomograf ini menggunakan motor penggerak. Gerakan tabung ini memiliki kecepatan bervariasi. Pancaran sinar-x terjadi selama gerakan tabung dari posisi awal hingga posisi akhir. Waktu gerakan yang cepat dibutuhkan untuk menghasilkan gambar yang teliti pada obyek bergerak seperti paru-paru dan jantung sekaligus untuk mengurangi paparan radiasi sinar-x.

Gerakan tabung yang membutuhkan waktu lama bisa menyebabkan perubahan posisi pasien atau posisi obyek di tengah *ekspose*, yang menyebabkan posisi fokus pada obyek berubah , pada akhirnya hasil gambar yang diperoleh tidak jelas. Juga paparan radiasi sinar-x yang lama dapat menimbulkan bahaya kesehatan. Peralatan tomograf yang ada saat ini membutuhkan waktu yang relatif lama dalam pengambilan gambar.

1.2 Permasalahan

Bagaimana menciptakan kontrol gerakan tabung tomograf yang cepat dan tepat untuk mempersingkat waktu *ekspose* yang dibutuhkan sehingga diperoleh hasil gambar yang jelas dan benar, serta dapat mengurangi besarnya paparan radiasi sinar-x yang mengenai pasien.

1.3 Tujuan

Tujuan tugas akhir ini adalah :

1. Memperdalam pemahaman dan pengetahuan tentang logika *fuzzy* dan kontrol *fuzzy*.
2. Memperdalam pemahaman dan pengetahuan tentang mikrokontroler dan penerapannya sebagai kontrol *fuzzy*
3. Merancang dan membuat kontrol gerakan tabung tomograf menggunakan kontrol *fuzzy*
4. memasyarakatkan dan memperluas penggunaan kontrol *fuzzy* dalam bidang kedokteran.

1.4 Pembatasan Masalah

Dalam tugas akhir ini akan digunakan resistor variabel sebagai sensor posisi. Keluaran dari resistor variabel diubah dalam bentuk tegangan, kemudian dikonversi ke dalam bentuk data digital menggunakan ADC MAX154. Kontroler yang dipergunakan adalah mikrokontroler AT89C51. Metode *fuzzy* digunakan dalam pengaturan posisi dan pengaturan kecepatan putar motor arus searah.

Masukan kontroler *fuzzy* berupa kesalahan (*Error*) posisi dan perubahan kesalahan (*DError*). Masukan ini akan menentukan keluaran *fuzzy* yang akan mengatur kecepatan pergeseran motor menuju posisi yang diinginkan.

1.5 Metodologi

Pengerjaan tugas akhir ini dilakukan dengan langkah-langkah terlebih dahulu studi literatur tentang logika *fuzzy*, kontrol *fuzzy*, mikrokontroler, dan studi tentang peralatan tomograf serta teori penunjang lainnya. Perancangan dan pembuatan perangkat keras berupa mikrokontroler, *Analog to Digital Converter* (ADC), *driver* motor. Kemudian dilanjutkan dengan perancangan peralatan mekanis.

Bersamaan dengan perancangan alat, dilakukan pula perancangan perangkat lunak. Dan keseluruhan perangkat keras dan perangkat lunak disatukan untuk membentuk sistem pengaturan gerakan tabung sinar-x pada peralatan tomograf. Setelah itu dilakukan pengujian dan kalibrasi alat serta pengambilan data dari alat yang telah dibuat. Kemudian dilanjutkan dengan penyusunan laporan tugas akhir.

1.6 Sistematika

Sistematika penulisan buku tugas akhir ini dibagi dalam beberapa bab berikut ini :

BAB I : Pendahuluan, menguraikan latar belakang, permasalahan, tujuan, pembatasan masalah, metodologi, sistematika, dan relevansi tugas akhir.

BAB II : Teori Penunjang, membahas mengenai teori yang mendasari sistem dan komponen yang digunakan dalam tugas akhir ini.

BAB III : Perancangan alat, membahas perancangan sistem yang akan dibuat yang meliputi perancangan mekanik, perangkat keras, dan perangkat lunak.

BAB IV : Pengujian dan pengukuran, menyajikan hasil-hasil yang diperoleh dalam pengujian dan pengukuran sistem yang telah dibuat.

BAB V : Penutup, berisi kesimpulan tugas akhir dan saran-saran.

1.7 Relevansi

Diharapkan dari tugas akhir ini dapat memberikan sumbangan untuk mengembangkan dan menerapkan lebih jauh kontrol *fuzzy*, khususnya dalam bidang instrumentasi medika dan industri pada umumnya.

BAB II

DASAR TEORI

2.1 Logika *Fuzzy*

Konsep teori *fuzzy* pertama kali diperkenalkan oleh L.A. Zadeh pada tahun 1965 berupa teori Himpunan *Fuzzy* (*fuzzy set*). Konsep himpunan *fuzzy* ini dilatarbelakangi oleh kebutuhan metode yang merepresentasikan dan menganalisa fenomena-fenomena di alam nyata yang serba tidak tepat ditinjau dari cara berfikir manusia terhadap fenomena-fenomena itu.

Cara berfikir manusia pada dasarnya tidak hanya dalam bilangan-bilangan yang pasti, tetapi merupakan penggolongan-penggolongan dengan batasan-batasan yang samar. Logika manusia dalam memandang suatu fenomena tidak hanya terdiri atas dua nilai (1 dan 0), tetapi merupakan logika multi nilai yang berangsur-angsur (*gradual*) antara suatu penggolongan dengan penggolongan lainnya. Berdasarkan penggolongan tersebut ada hubungan dengan aturan yang samar dalam proses berfikir manusia dalam mengambil keputusan untuk menghadapi fenomena-fenomena itu.

2.1.1 Himpunan *Fuzzy*

Suatu himpunan *fuzzy* F dalam semesta X didefinisikan sebagai kumpulan pasangan elemen x dan fungsi keanggotaan $\mu_F(x)$. Fungsi keanggotaan $\mu_F(x)$ mempunyai nilai dalam interval [0,1] pada tiap x dalam X. Nilai fungsi keanggotaan menunjukkan tingkat keanggotaan elemen x dalam F. Bila $\mu_F(x) = 1$, x merupakan anggota penuh F, sedang $\mu_F(x) = 0$ menunjukkan x bukan anggota F.

Secara Umum himpunan *fuzzy* dinotasikan sebagai

$$F = \{(x,\mu_F(x)) \mid x \in X\} \quad (2.1)$$

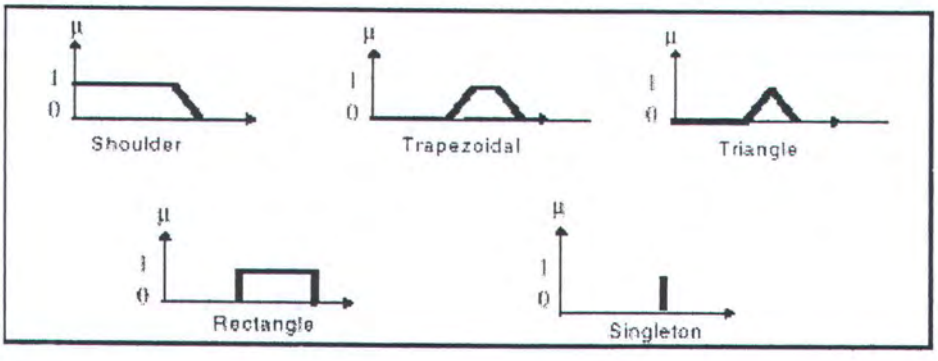
Bila x diskrit dengan n elemen, F dapat dinyatakan sebagai

$$F = \sum_{i=1}^{n} \mu_F(x_i)|x_I \quad (2.2)$$

Bila x kontinu, F dapat dinyatakan sebagai

$$F = \int_x \mu_F(x)|x \quad (2.3)$$

2.1.2 Fungsi Keanggotaan Himpunan *Fuzzy*



Gambar 2.1 Bentuk-bentuk fungsi keanggotaan yang sering digunakan

Secara umum ada dua metode untuk menyatakan suatu fungsi keanggotaan yaitu pendefinisian secara numerik dan bentuk fungsi. Fungsi keanggotaan yang didefinisikan secara numerik menggunakan pendukung diskrit. Sedangkan bentuk fungsi digunakan pada pendukung kontinu. Gambar 2.1 menunjukkan bentuk-bentuk umum fungsi keanggotaan.

Contoh fungsi keanggotaan yang sering digunakan antara lain,

- Fungsi Eksponensial
- Fungsi Segitiga

$$\mu_F(x) = 1 - \frac{\sqrt{(x-a)^2}}{b} \tag{2.4}$$

- Fungsi Trapesium

$$\mu_F(x) = \begin{cases} 1 & ;0 \le (x-a) \le \frac{b}{2} \\ 2 - 2\frac{\sqrt{(x-a)^2}}{b} & ;\frac{b}{2} \le (x-a) \le b \end{cases} \tag{2.5}$$

2.1.3 Operasi Dasar Himpunan *Fuzzy*

Hubungan-hubungan berikut diterapkan antara dua himpunan *fuzzy*, dimana elemen x berasal dari himpunan dasar G

- *equality* A = B,

  benar jika $\mu_A(x) = \mu_B(x)$, untuk semua $x \in G$

- *complete inclusion* $A \subseteq B$

  benar jika $\mu_A(x) \le \mu_B(x)$, untuk semua $x \in G$

- *partial inclution* $A \subset B$

  benar jika $\mu_A(x) \le \mu_B(x)$, untuk semua $x \in G$

  dan $\mu_A(x) < \mu_B(x)$, untuk sebuah $x \in G$

Operasi-operasi berikut dapat diterapkan antara dua himpunan *fuzzy*, dimana elemen x berasal dari himpunan dasar G

- irisan (*intersection*)

  $A \cap B$ didefinisikan dengan $\mu_{A\cap B}(x) = I(\mu_A(x),\mu_B(x))$,

  dimana I disebut sebagai operator irisan.

- gabungan (*union*)

  $A \cup B$ didefinisikan dengan $\mu_{A\cup B}(x) = U(\mu_A(x),\mu_B(x))$,

  dimana U disebut sebagai operator gabungan

- komplemen (*complement*)

  didefinisikan dengan $\mu_{\bar{A}}(x) = 1 - (\mu_A(x)$

seperti juga pada himpunan klasik, interpretasi di bawah ini diterapkan pada himpunan *fuzzy* :

- irisan bersesuaian dengan operator logika AND
- gabungan bersesuaian dengan operator logika OR
- komplemen bersesuaian dengan operator logika NOT

Algoritma dasar untuk implementasi matematis dari irisan, gabungan, dan komplemen adalah sebagai berikut :

- untuk irisan adalah minimum

  $\mu_{A\cap B}(x) = \text{Min}(\mu_A(x),\mu_B(x))$, $x \in G$ (2.6)

- untuk gabungan adalah maksimum

  $\mu_{A\cup B}(x) = \text{Max}(\mu_A(x),\mu_B(x))$, $x \in G$ (2.7)

- untuk komplemen adalah pengurangan dari satu.

  $\mu_A(x) = 1 - (\mu_A(x)$, $x \in G$ (2.8)

Sebagai tambahan dari operator logika *fuzzy* dasar tersebut di atas, masih banyak pilihan untuk operator *fuzzy* lainnya, terutama untuk aplikasi non kontrol.

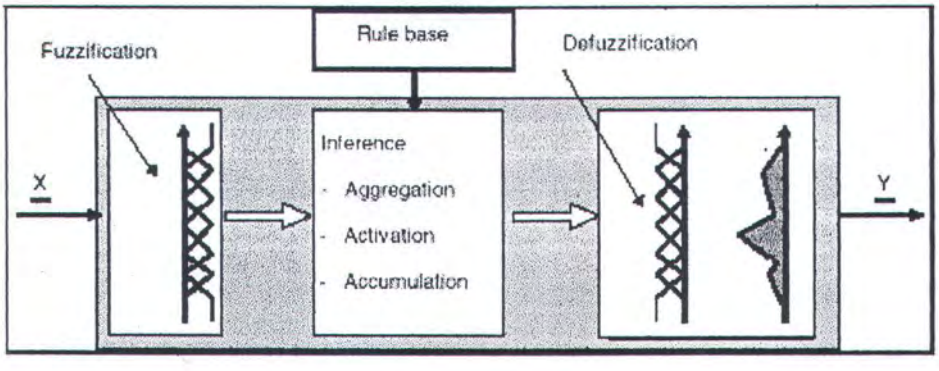
2.2 Kontrol *Fuzzy*[1]

Kontrol *fuzzy* (*fuzzy control*) adalah kontrol *open* dan *close loop* dari proses teknis, termasuk proses dari nilai terukur, yang didasarkan pada penggunaan aturan *fuzzy* dan prosesnya dengan bantuan logika *fuzzy*.

Informasi masukan terdiri dari variabel nyata dalam bentuk variabel proses yang dapat diukur, *derived variables*, dan *set point*. Variabel keluaran dalam bentuk variabel terkoreksi. Transformasi harus ditunjukkan antara masukan dan keluaran dari proses dan bidang *fuzzy* (fuzzifikasi,defuzzifikasi). Komponen utama dari kontrol *fuzzy* terdiri atas aturan linguistik dari *rule base* dan *inference*.

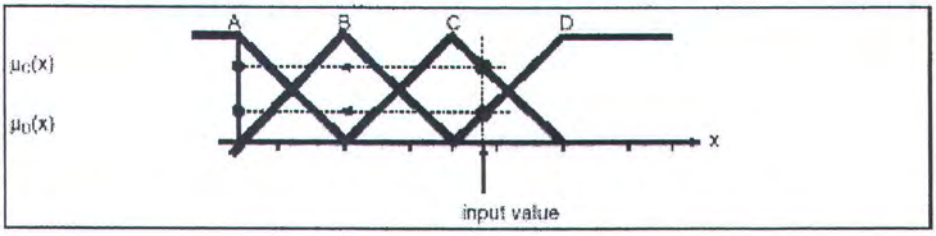
2.2.1 Fuzzifikasi

Pemetaan yang memetakan masukan yang bersifat *crisp* (bukan *fuzzy*) ke semesta himpunan *fuzzy* disebut fuzzifikasi. Di sini, variabel masukan ditentukan derajat keanggotaannya pada setiap *linguistic term* dari *linguistic variable* yang sesuai.



Gambar 2.2 Struktur dan elemen fungsional dari kontrol *fuzzy*

[1] IEC, *International Standard IEC 1131 - Programmable Controllers, Part 7 - Fuzzy Control Programming.* Committee Draft CD 1.0 (Rel. 19 Jan 97) (Frankfurt : international Task Force (TF8) of IEC/65B/WG7, 1996), pp.29-36.



Gambar2.3 Prinsip fuzzifikasi

2.2.2 *Rule Base*

Rule base berisi pengetahuan empiris yang mengacu pada operasi dari proses khusus. Aturan linguistik digunakan untuk merepresentasikan pengetahuan. Ini dapat dengan mudah ditunjukkan bahwa sebuah aturan *fuzzy* R_j yang berdasar pada kombinasi OR dari m pernyataan dapat direpresentasikan dengan m aturan, dimana pernyataan-pernyataan hanya dikombinasikan dengan AND. Gambar 2.4 dan 2.5 merupakan contoh dari representasi aturan yang berbeda.

Rule Base

- Definition of the rule base in textual form

R1: IF Condition P1 THEN Conclusion C1
R2: IF Condition P2 THEN Conclusion C2
R3: IF Condition P3 THEN Conclusion C3
...
R_n: IF Condition P_n THEN Conclusion C_n

where condition can consist of a combination of linguistic statements, e.g. for rule k

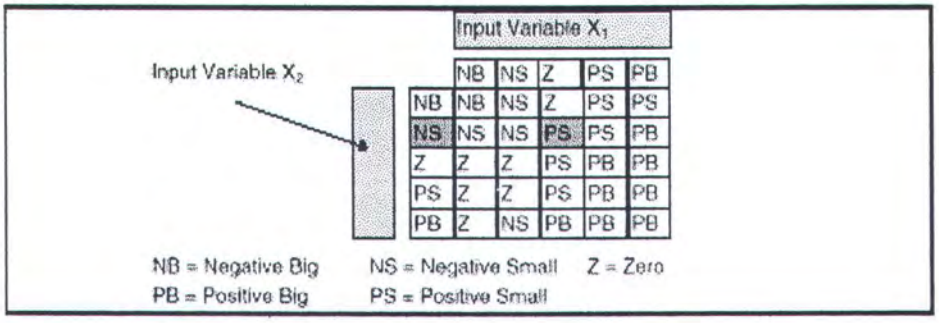
$P_k = P_{k1}$ AND P_{k2} OR (NOT P_{k3})

and for conclusion there may be more than one, e.g.

$C_k = C_{k1}, C_{k2}$

Gambar 2.4 Representasi dari dasar pengetahuan dalam bentuk aturan linguistik

Jika dua variabel masukan dan satu variabel keluaran tersedia, dan jika dua variabel ini hanya dikombinasikan dengan operator AND, *rule base* dapat diberikan dalam bentuk matriks, dimana nilai dari variabel masukan diberikan pada kolom dan/atau garis, dan matriks berisi nilai variabel keluaran.



Gambar 2.5 Representasi matriks dari dua variabel

2.2.3 *Inference*

Execution of the rule base broken down into the following steps:

- Aggregation
  Determing the degree of accomplishement of the condition
  from the degree of membership of the subconditions

  P_{k1} AND P_{k2} OR (NOT P_{k3}) = P_k
  for all n conditions

- Activation
  Activation of the IF-THEN conclusion

  If condition P_k THEN conclusion C_k
  for all n rules

  - Consideration of the weighting factor of each rule

- Accumulation
  Combination of the weighted results
  of the rules into an overall result

Gambar 2.6 Elemen-elemen *Inference*

Definisi yang lebih umum dan pembuktian matematis dari *rule base fuzzy* didasarkan pada *modus ponen* tergeneralisasi dan prinsip implikasi *fuzzy*. Prinsip dan definisi yang digunakan di sini berlaku untuk kasus sederhana dari *rule base fuzzy*, pada dasarnya muncul inspirasi dari luasnya penggunaan pola *inference* Mamdani. Lebih banyak pola *inference* yang lebih sulit tidak dalam batas-batas standar ini. *Inference* terdiri atas tiga subfungsi *agregation*, *activation*, dan *accumulation* ditunjukkan pada gambar 2.6.

- *Agregation* :

Jika kondisi berisi satu subkondisi, maka keabsahan dari kondisi tersebut identik dengan kondisi itu, misal derajat pencapaian kondisi sesuai dengan kondisi. Bagaimanapun juga, jika kondisi berisi kombinasi dari beberapa subkondisi, derajat pencapaian harus ditentukan dengan *aggregation* dari nilai-nilai individual. Jika ada kombinasi AND dari subkondisi, derajat pencapaian dihitung dengan memakai operator logika *fuzzy* AND.

- *Activation*

Pada konklusi, subkonklusi berhubungan dengan variabel keluaran. Derajat keanggotaan konklusi ditentukan pada basis derajat pencapaian kondisi yang ditentukan dalam *aggregation* (konklusi IF A THEN B). Pada umumnya, MIN atau PROD digunakan untuk *activation*.

Jika *rule base* berisi aturan-aturan dengan faktor bobot w_k, dimana $w_k \in [0,1]$, ini dapat diterapkan dengan memakai perkalian.

$$a_k^* = w_k \times a_k \qquad (2.9)$$

- *Accumulation*

Hasil dari aturan dikombinasikan untuk memperoleh hasil keseluruhan. Algoritma maksimum biasanya digunakan untuk *accumulation*. Tabel 2.1 menunjukkan operator yang umumnya digunakan untuk langkah-langkah *inference* individual.

Tergantung pada kombinasi operator pada langkah individual, didapatkan strategi *inference* yang berbeda. Pengetahuan terbaik dinamakan *MaxMin inference* dan *MaxProd inference* , yang menggunakan maksimum untuk *accumulation* dan minimum atau produk aljabar untuk *activation*. Dalam kasus *MaxMin inference*, fungsi keanggotaan himpunan *fuzzy* dari konklusi dibatasi pada derajat pencapaian kondisi, dan kemudian dikombinasikan untuk menciptakan himpunan *fuzzy* dengan membentuk maksimum. Dalam *MaxProd inference*, sebaliknya, fungsi keanggotaan himpunan *fuzzy* dari konklusi diberi bobot, misal perkalian, dengan derajat pencapaian dari kondisi dan kemudian dikombinasikan.

Tabel 2.1 Langkah-langkah *inference* dan algoritma yang umum digunakan

Langkah Inference	***Operator***	***Algoritma***
Aggregation		
untuk AND	*Minimum*	$a_k = \text{Min}\ \{a_{K1}(x), a_{K2}(x)\}$
untuk OR	*Maksimum*	$a_k = \text{Max}\ \{a_{K1}(x), a_{K2}(x)\}$
Activation		
konversi dari konklusi IF-THEN		
	Minimum	$c_k' = \text{Min}\ \{a_K, \mu_{\alpha}(u)\}$
Faktor bobot dari setiap aturan		
	Perkalian	$c_k = \text{Mult}\{\omega_K, c_K'\} = \omega_K \times c_K'$
Accumulation	*Maksimum*	$\mu_{accu}(u) = \text{MAX}\ \{c_i(u)\}$

2.2.4 Defuzzifikasi

Inference mensuplai sebuah himpunan *fuzzy* atau fungsi keanggotaannya sebagai hasil. Sebuah elemen kontrol tidak dapat secara langsung memproses informasi *fuzzy* ini sehingga hasil proses *inference* diubah menjadi nilai numerik *crisp*. Dalam konteks ini, nilai *crisp* yang ditentukan (biasanya angka real) harus menyediakan representasi yang bagus dari informasi yang ada di dalam himpunan *fuzzy*.

Beberapa metode defuzzifikasi yang digunakan :

- Metode *Center of Gravity* (CoG)

  Variabel keluaran *crisp* ditentukan sebagai nilai *abscissa*.

$$U = \frac{\int_{Min}^{Max} u * \mu(u) * du}{\int_{Min}^{Max} \mu(u) * du} \tag{2.10}$$

  Jika fungsi keanggotaan nilai keluaran berupa *singleton*, perhitungan diberikan secara *Center of Gravity Method for Singleton* (COGS).

$$U(t_k) = \frac{\sum_{i=1}^{p} [U_i * a_i(t_k)]}{\sum_{i=1}^{p} [a_i(t_k)]} \tag{2.11}$$

- *Left Most Maximum* LM

  Nilai dari variabel keluaran ditentukan di mana fungsi keanggotaan keluaran mencapai maksimum paling kiri.

- *Right Most Maximum* RM

  Nilai dari variabel keluaran ditentukan di mana fungsi keanggotaan keluaran mencapai maksimum paling kanan.

- Metode *Centre of Area* (CoA)

  Di sini, nilai keluaran ditentukan sebagai nilai *abscissa* dari pusat dimana membagi luasan di bawah fungsi keanggotaan menjadi 2 luasan yang sama.

2.3 Pengontrolan dengan Logika *Fuzzy*

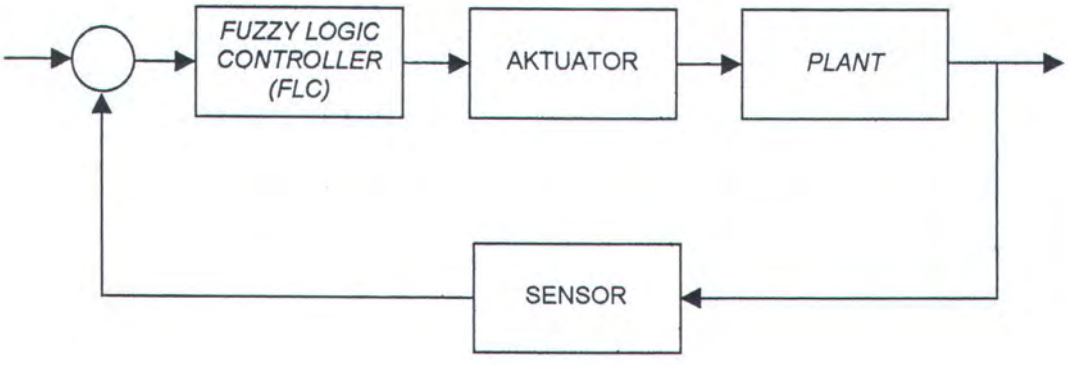
Tujuan utama suatu sistem kontrol adalah menghasilkan suatu keluaran yang dikehendaki untuk setiap masukan yang diberikan. Rangkaian usaha yang dilakukan untuk mengolah masukan menjadi keluaran yang dikehendaki disebut dengan proses kontrol. Secara konvensional dikenal beberapa proses kontrol yaitu metode *look-up table*, metode pemodelan secara matematis untuk mencari fungsi transfer antara masukan dan keluaran, dan metode dengan menggunakan logika *fuzzy*.

Model *look-up table* efektif hanya jika sistem yang dikontrol memiliki resolusi rendah dan variabel masukan sedikit. Kelemahan utama sistem ini adalah jika sistem rumit maka akan menghabiskan banyak tempat di memori, menimbulkan gangguan pada sistem karena adanya *event* yang tidak tertanggapi, dan meloncatnya nilai keluaran dari tabel ke tabel yang lainnya.

Model matematis dari suatu sistem harus secara tepat menggambarkan perilaku sistem terhadap masukan yang diberikan. Untuk sistem yang rumit, hal ini menjadikan pembuatan model matematis amat sulit, dengan hasil suatu persamaan yang kompleks dan sulit diaplikasikan. Pembuatan model matematis juga memerlukan keahlian khusus sehingga meningkatkan biaya desain. Kelemahan lain dari model ini adalah keterbatasan data yang ada sehingga tidak memungkinkan pembuatan model matematis dan mengurangi keakuratan sistem.

Penggunaan teknologi *fuzzy* dalam rekayasa proses dan sistem informasi akan menghasilkan alat-alat yang handal, tahan, luwes dan lebih canggih dibandingkan dengan alat-alat digital biasa. Hal ini akan memproduksi sistem pengambilan keputusan, sistem kontrol otomatis yang akan membawa kepada mesin yang memiliki daya pikir.

Struktur dasar penggunaan logika *fuzzy* untuk sistem kontrol ditunjukkan seperti gambar 2.7.



Gambar 2.7 Diagram blok sistem kontrol logika *fuzzy*

Sistem kontrol otomatis biasanya terdiri atas empat bagian utama, yaitu sensor, kontroler, aktuator, dan *plant*. Sensor berfungsi sebagai pengambil data perilaku sistem. Aktuator memberikan daya untuk menggerakkan *plant* (yaitu peralatan yang dikontrol), sehingga dicapai suatu harga yang diinginkan. Kontroler berfungsi memberikan sinyal perintah ke aktuator sesuai aksi kontrol menurut besarnya deviasi (*error*) yaitu selisih antara referensi dan keluaran yang terukur oleh sensor.

Sistem kontrol yang digunakan adalah sistem kontrol dengan umpan balik[2] karena sistem ini cenderung menjaga hubungan yang telah ditentukan antara keluaran dan masukan acuan dengan membandingkannya dan menggunakan selisihnya sebagai alat pengontrolan.

2.4 Mikrokontroler AT89C51

2.4.1 Konfigurasi AT89C51

- *Compatible* dengan MCS-51™
- *Reprogrammable Flash Memory 4 Kbyte* internal
  (± 1000 kali program ulang)
- *Full* operasi statik : 0 Hz s.d. 24 MHz
- *Program Memory Lock* tiga tingkat
- RAM internal 128 x 8-bit
- 32 *line Programmable I/O*
- Dua *Timer/Counter* 16-bit
- Enam *Interrupt Sources*
- Kanal *Programmable Serial*
- Mode *Low Power Idle* dan *Power Down*

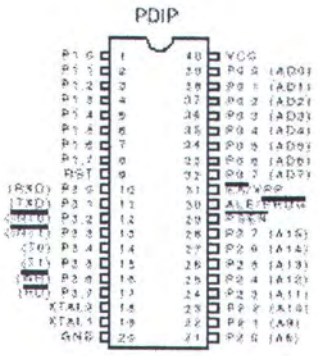
[2] Katsuhito Ogata, Edi Leksono, *Teknik Kontrol Automatik Jilid 1* (Jakarta : Erlangga, 1991), p.3.

AT89C51 memiliki konfigurasi standar : 4 *Kbyte* Flash ROM, 128 *byte* RAM, 32 *line programmable* I/O, *timer/counter* 16-bit, lima vektor arsitektur interupsi dua-tingkat, *port* serial *full duplex*, osilator internal dan *clock*. Konfigurasi tambahan, AT89C51 dirancang menggunakan *static logic* untuk operasi sampai dengan frekuensi nol dan tersedia dua mode *software selectable power saving*. Mode *idle* menghentikan CPU, tetap memfungsikan RAM, *timer/counter*, *port* serial, dan sistem interupsi. Mode *power down* menyimpan isi RAM tetapi menghentikan osilator sehingga menghentikan semua fungsi *chip* lainnya sampai *hardware* di-*reset*.

2.4.2 Deskripsi AT89C51

AT89C51 berdaya rendah, merupakan mikrokomputer 8-bit *high performance* CMOS dengan 4 *Kbyte Flash Programmable and Erasable Read Only Memory* (PEROM). Alat ini dirancang menggunakan teknologi Atmel *high density nonvolatile memory* yang *set* instruksi dan kaki-kakinya *compatible* dengan standar industri MCS-51™. *Flash memory* internal memudahkan pemrograman ulang dalam sistem maupun secara konvensional. Dengan menggabungkan CPU 8-bit serbaguna dengan *flash memory* pada sebuah *chip*, Atmel AT89C51 merupakan mikrokomputer yang sangat tangguh yang menyediakan fleksibilitas yang tinggi dan pemecahan efektifitas biaya pada banyak aplikasi kontrol.

2.4.3 Konfigurasi Kaki AT89C51



Gambar 2.8 Konfigurasi kaki Atmel AT89C51

V_{cc}

Suplai tegangan.

GND

Ground.

Port 0

Port 0 adalah sebuah *port* I/O *open drain bidirectional* 8-bit. Keluaran setiap kaki dapat menahan 8 buah masukan TTL. Saat logika '1' diberikan pada kaki-kaki *port* 0, maka kaki-kaki tersebut dapat digunakan sebagai masukan *high-impedance*.

Port 0 bisa juga dikonfigurasikan untuk multipleks *low-order address* dan data selama mengakses program dan data memori eksternal. Pada mode ini P0 memiliki *pull-up* internal. *Port* 0 juga menerima *byte* kode selama *flash programming* dan mengeluarkan *byte* kode selama verifikasi program. Diperlukan *pull-up* eksternal selama verifikasi program.

Port 1

Port 1 adalah sebuah *port* I/O *open drain bidirectional* 8-bit dengan *pull-up* internal. *Buffer* keluaran *port* 1 dapat menahan 4 masukan TTL. Ketika logika '1' ditulis pada kaki *port* 1, kaki tersebut di-*pull high* oleh *pull-up* internal dan dapat digunakan sebagai masukan. Sebagai masukan, kaki *port* 1 yang di-*pull low* secara eksternal akan menghasilkan arus (I_{IL}) karena *pull-up* internal. *Port* 1 juga menerima byte *low-order address* selama *flash programming* dan verifikasi program.

Port 2

Port 2 adalah sebuah *port* I/O *open drain bidirectional* 8-bit dengan *pull-up* internal. *Buffer* keluaran *port* 2 dapat menahan 4 masukan TTL. Ketika logika '1' ditulis pada kaki *port* 2, kaki tersebut di-*pull high* oleh *pull-up* internal dan dapat digunakan sebagai masukan. Sebagai masukan, kaki *port* 2 yang di-*pull low* secara eksternal akan menghasilkan arus (I_{IL}) karena *pull-up* internal.

Port 2 mengeluarkan byte *high-order address* selama pengambilan memori program eksternal dan selama mengakses memori data eksternal yang menggunakan alamat 16-bit (MOVX @DPTR). Pada aplikasi ini *port* 2 menggunakan *pull-up* internal ketika menghasilkan logika '1'. Selama akses ke memori data eksternal yang menggunakan alamat 8-bit (MOVX @RI), *port* 2 mengeluarkan isi dari *Special Function Register* P2. *Port* 2 juga menerima bit-bit *high-order address* dan beberapa sinyal kontrol selama *flash programming* dan verifikasi program.

***Port* 3**

Port 3 adalah sebuah *port* I/O *open drain bidirectional* 8-bit dengan *pull-up* internal. *Buffer* keluaran *port* 3 dapat menahan 4 masukan TTL. Ketika logika '1' ditulis pada kaki *port* 1, kaki tersebut di-*pull-high* oleh *pull-up* internal dan dapat digunakan sebagai masukan. Sebagai masukan, kaki *port* 3 yang di-*pull low* secara eksternal akan menghasilkan arus (I_{IL}) karena *pull-up* internal. *Port* 3 juga menerima beberapa sinyal kontrol untuk *flash programming* dan verifikasi program.

Port 3 juga melayani fungsi-fungsi lainnya sebagai berikut :

Tabel 2.2 Fungsi alternatif *port* 3

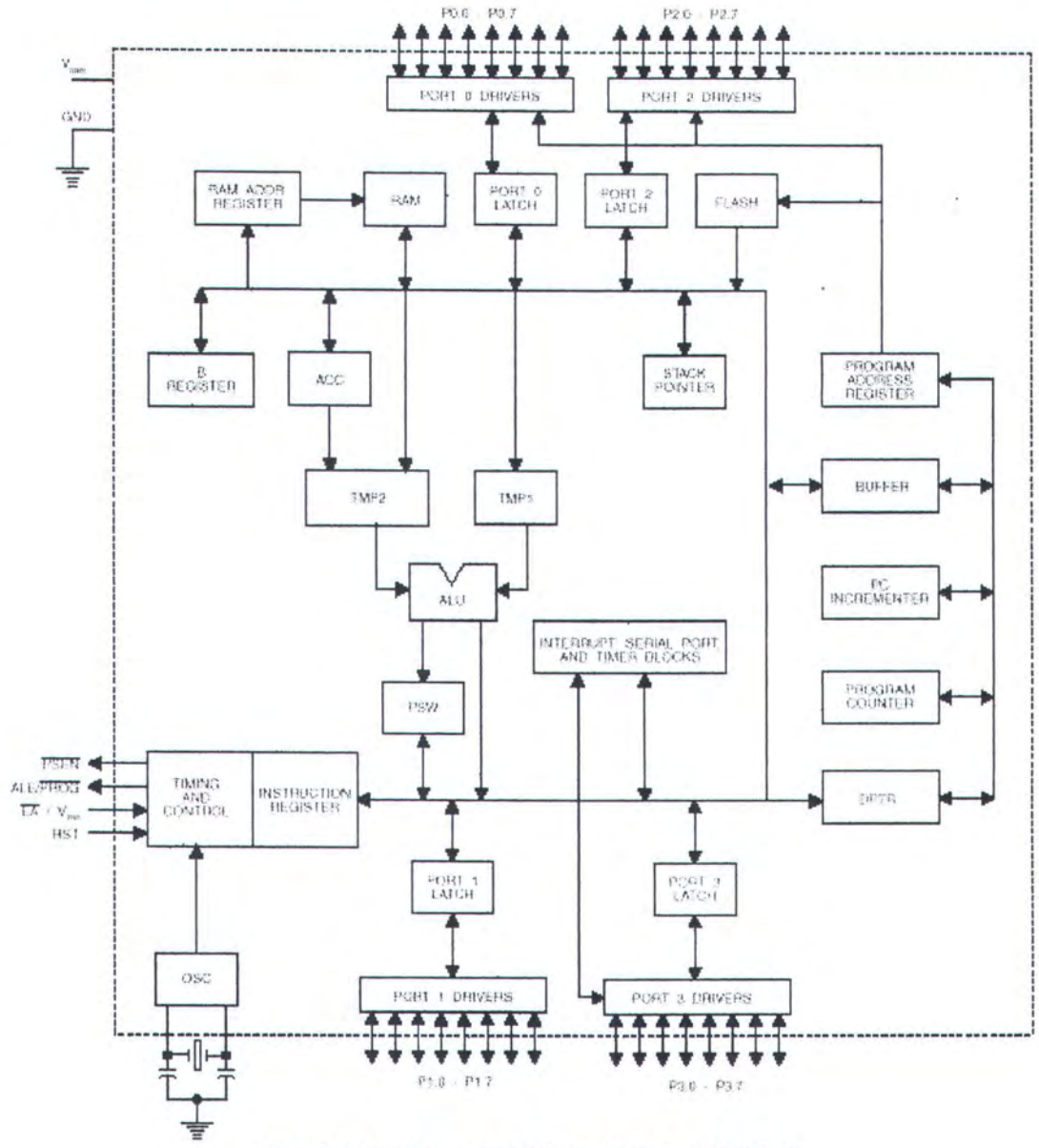
Kaki *Port*	Fungsi Alternatif
P3.0	RXD (*port* Serial Input)
P3.1	TXD (*port* Serial Output)
P3.2	INT0 (*External Interrupt* 0)
P3.3	INT1 (*External Interrupt* 1)
P3.4	T0 (*Timer 0 External Input*)
P3.5	T1 (*Timer 1 External Input*)
P3.6	WR (*external data memory write strobe*)
P3.7	RD (*external data memory read strobe*)

RST

Reset input. Ketika osilator bekerja, sinyal *high* pada kaki ini selama dua siklus mesin akan me-*reset* alat ini.

ALE/PROG

Address Latch Enable mengeluarkan pulsa untuk menahan *byte* rendah dari alamat selama mengakses memori eksternal. Kaki ini juga sebagai masukan pulsa *proggram* (PROG) selama *flash programming*.



Gambar 2.9 Diagram Blok Atmel AT89C51

Dalam operasi normal, ALE mengeluarkan 1,6 kali frekuensi osilator secara konstan, dan bisa digunakan untuk pewaktu eksternal atau berfungsi sebagai *clock*. Dengan catatan 1 pulsa terabaikan selama mengakses memori data eksternal.

Bila diinginkan, Operasi ALE dapat di-*disable* dengan men-*set* bit 0 lokasi 8EH pada SFR. Dengan men-*set* bit tersebut, ALE akan aktif hanya pada saat instruksi MOVX atau MOVC. Jika tidak, kaki ini di-*pull high* dengan lemah.

Penge-*set*-an bit ALE-*disable* tidak menimbulkan efek bila mikrokontroler dalam mode eksekusi eksternal.

PSEN

Program Store Enable adalah sebuah *read strobe* untuk memori program eksternal. Pada saat AT89C51 mengeksekusi kode dari memori program eksternal, PSEN diaktifkan dua kali pada setiap siklus mesin, kecuali aktivasi 2 PSEN itu diabaikan selama tiap kali akses ke memori data eksternal.

EA/V $_{PP}$

External Access Enable. EA harus dihubungkan ke GND supaya mengaktifkan AT89C51 untuk mengambil kode dari memori program eksternal yang berada pada lokasi 0000H sampai dengan FFFFH. Bagaimanapun juga apabila *lock bit* 1 diprogram, EA akan ditahan secara internal pada saat *reset*. EA harus dihubungkan pada V_{CC} untuk eksekusi progran internal. Kaki ini juga menerima tegangan *programming enable* 12 volt (V_{PP}) selama *flash programming* untuk IC yang memerlukan V_{PP} 12 volt.

XTAL1

Masukan ke *inverting oscillator amplifier* dan masukan ke *internal clock operating circuit*.

XTAL2

Keluaran dari *inverting oscillator amplifier*.

2.4.4 Organisasi Memori [3]

Semua mikrokontroler keluarga MCS-51 memiliki pembagian ruang alamat (*address space*) untuk program dan data secara terpisah. Maksimum memori program dan data yang masih dapat diakses masing-masing sejumlah 64K alamat.

Memori program hanya dapat dibaca saja, sementara memori data bisa dibaca dan ditulis. Pembacaan memori program eksternal dengan kaki EA=0 dan kaki EA=1 untuk memori program internal. Sinyal yang membolehkan pembacaan memori program ekternal adalah kaki PSEN (*program store enable*), sedang untuk pembacaan memori data eksternal, CPU mngirim sinyal READ dan WRITE.

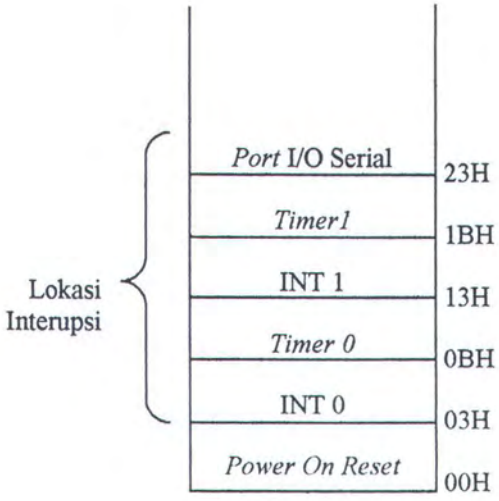
Mikrokontroler MCS-51 memiliki 5 buah ruang alamat, yaitu :

1. Ruang alamat kode sebanyak 64K yang seluruhnya merupakan ruang alamat kode eksternal (*off-chip*).
2. Ruang alamat data internal yang dapat dialamati secara langsung, yang terdiri atas RAM sebanyak 128 *byte* dan *hardware register* sebanyak 128 *byte*
3. Ruang alamat data internal yang dialamati secara tidak langsung sebanyak 128 *byte*, seluruhnya diakses dengan pengalamatan tidak langsung.
4. Ruang alamat data eksternal sebanyak 64K *byte* (*off-chip*) yang dapat ditambahkan oleh pemakai.
5. Ruang alamat bit dapat diakses dengan pengalamatan bit langsung.

---

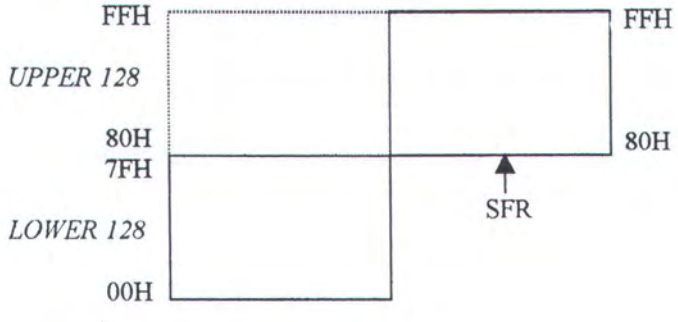
[3] Moh. Ibnu Malik dan Anistardi, *Bereksperimen dengan Mikrokontroler* (Jakarta : PT. Elex Media Komputindo, 1997), pp.9-12

Bagian bawah memori program digunakan untuk alokasi instruksi interupsi yang ada. Setiap interupsi memiliki lokasi tetap dalam memori program.



Gambar 2.10 Alokasi interupsi pada memori program

Memori data internal dipetakan seperti gambar 2.11. Ruang memorinya dibagi menjadi 3 blok, yaitu *lower* 128, *upper* 128, dan ruang *special function register* (SFR) 128.



Gambar 2.11 Memori data internal

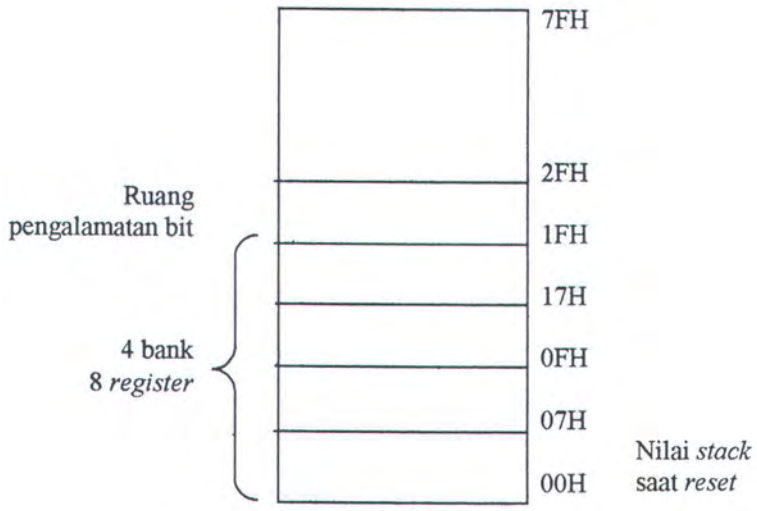
SFR berisi *register-register* dangan fungsi tertentu. Tabel 2.3 merupakan nama-nama, simbol dan alamat dari SFR.

Tabel 2.3 *Special Function Register* [4]

Simbol	Nama	Alamat
ACC	Akumulator	E0H
B	*Register* B	F0H
PSW	*Program Status Word*	D0H
SP	*Stack Pointer*	81H
DPTR	*Data Pointer* 16 bit	
	DPL *byte* rendah	82H
	DPH *byte* tinggi	83H
P0	*Port* 0	80H
P1	*Port* 1	90H
P2	*Port* 2	A0H
P3	*Port* 3	B0H
IP	*Interrupt Priority Control*	B8H
IE	*Interrupt Enable Control*	A8H
TMOD	*Timer/Counter Mode Control*	89H
TCON	*Timer/Counter Control*	88H
TH0	*Timer/Counter 0 High byte*	8CH
TL0	*Timer/Counter 0 Low byte*	8AH
TH1	*Timer/Counter 1 High byte*	8DH
TL1	*Timer/Counter 1 Low byte*	8BH
SCON	*Serial Control*	98H
SBUF	*Serial Buffer*	99H
PCON	*Power Control*	87H

Bagian bawah dari 128 *byte* dipetakan seperti gambar 2.12. Tiga puluh dua *byte* paling bawah dikelompokkan dalam 4 bank (8 *register*), yaitu R0 sampai R7. Dua bit dalam *program status word* (PSW) digunakan untuk memilih bank *register* yang digunakan.

[4] Ibid., p.19



Gambar 2.12. Bagian bawah 128 *byte* RAM internal

2.4.5 *Timer/Counter* [5]

Pada mikrokontroler AT89C51 terdapat 2 buah *timer/counter* 16 bit yang dapat diatur melalui perangkat lunak, yaitu *timer/counter* 0 dan *timer/counter* 1. Apabila *timer/counter* diaktifkan pada frekuensi kerja mikrokontroler 12 MHz, *timer* akan melakukan perhitungan waktu sekali setiap 1 μs secara bebas, tidak tergantung pada pelaksanaan suatu instruksi.

- sebagai *timer/counter* 8 bit

  $T = (255 - TLx) * 1\ \mu s$ (2.12)

  di mana TLx adalah isi *register* TL0 atau TL1

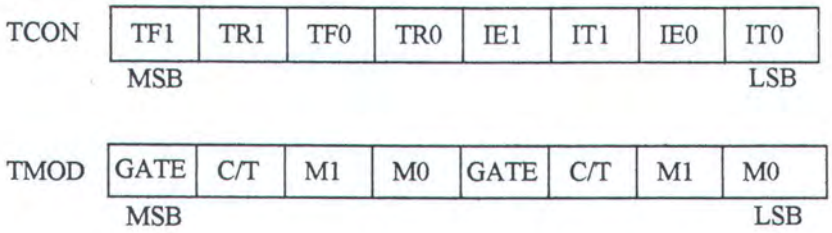
- sebagai *timer/counter* 16 bit

  $T = (65535 - THx\ TLx) * 1\ \mu s$ (2.13)

  TLx adalah isi *register* TL0 atau TL1

  THx adalah isi *register* TH0 atau TH1

[5] Ibid., pp.20-23



Gambar 2.13 Pengontrol kerja dan pemilih mode operasi *timer/counter*

Pengontrol kerja *timer/counter* adalah *register timer cotrol* (TCON), yang terdiri atas TFx yang merupakan *timer* x *overflow*, di-*set* oleh perangkat keras saat *timer/counter* menghasilkan limpahan (*overflow*), TRx adalah bit untuk menjalankan *timer* x, di-*set* oleh perangkat lunak untuk membuat *timer on* atau *off*, IEx eksternal *interrupt_x edge flag*, ITx *interrupt_x* tipe kontrol bit, *set* atau *clear* oleh perangkat lunak untuk menspesifikasikan sisi turun atau level rendah *trigger* dari interupsi eksternal.

Pengontrol pemilih mode operasi *timer/counter* adalah *register timer mode* (TMOD) yang terdiri atas GATE yang saat TRx di-*set* dan GATE=1, *timer/counter* x akan berjalan ketika TRx = 1 (*timer* dikontrol perangkat lunak), C/T adalah pemilih fungsi *timer* atau *counter*, *clear* untuk operasi *timer* dengan masukan dari sistem *clock* internal, *set* untuk operasi *counter* dengan masukan dari kaki T0 atau T1.

- Mode 0

  Register timer disusun sebagai *register* 13 bit. Setelah semua perhitungan selesai, mikrokontroler akan men-*set Timer Interrupt Flag* (TF1). Dengan membuat GATE = 1, *timer* dapat dikontrol oleh masukan luar INT1, untuk fasilitas pengukuran lebar pulsa.

- Mode 1

  Sama dengan mode 0 kecuali *register timer* akan bekerja dalam 16 bit.

- Mode 2

  Sebagai 8 bit *counter*, *overflow* dari TL1 tidak hanya men-*set* TF1, tetapi juga mengisi TL1 dengan TH1 diatur secara *software*, pengisian ini tidak merubah TH1.

- Mode 3

  Timer 1 dalam mode 3 hanya memegang hitungan. Efeknya sama dengan men-*set* TR1 = 0. *Timer* 0 dalam mode 3 menetapkan TL0 dan TH0 sebagai dua *counter* terpisah. TL0 menggunakan kontrol bit *timer* 0 yaitu C/T, GATE, TR0, INT0, dan TF0. TH0 ditetapkan sebagai fungsi *timer*.

  Mode 3 diperlukan untuk aplikasi yang memerlukan *timer/counter* ekstra 8 bit. Dengan *timer* 0 pada mode 3, seakan-akan memiliki 3 *timer/counter*. Saat *timer* 0 dalam mode 3, *timer* 1 dapat digunakan atau dimatikan, atau dapat digunakan oleh *port* serial sebagai pembangkit *baud rate*.

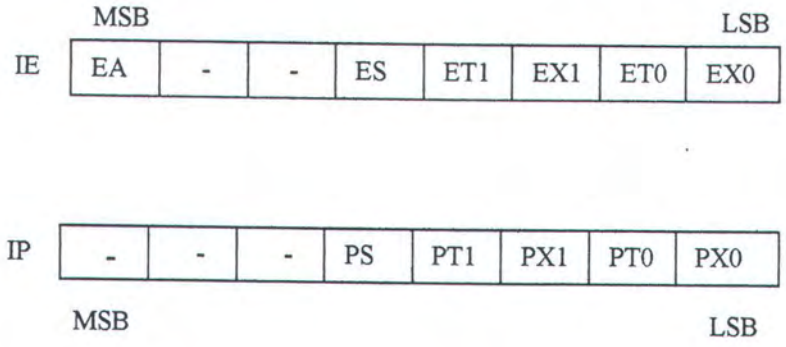
2.4.6 Interupsi [6]

Interupsi ada 2 jenis, yang tak dapat dihalangi oleh *software* (*non maskable interrupt*), contohnya *Reset*, dan interupsi yang dapat dihalangi oleh *software* (*maskable interrupt*, yang termasuk jenis ini adalah INT0 dan INT1 (eksternal), *timer/counter* 0, *timer/counter* 1, dan interupsi dari *port* serial (internal)).

---

[6] Ibid., pp.16-19

Ada 2 *register* yang mengontrol interupsi, yaitu IE (*interrupt enable*) dan IP (*interrupt priority*). AT89C51 tidak akan menanggapi permintaan interupsi jika suatu instruksi belum dilaksanakan secara lengkap (dapat selama 4 siklus).



Gambar 2.14 *Register Interrupt Enable (IE)* dan *Interrupt Priority (IP)*

Register IE memiliki bit-bit EA untuk melumpuhkan seluruh interupsi (EA=0) bila EA=1 maka setiap interupsi dapat digunakan atau dilumpuhkan secara individual, ES, ET1, EX1, ET0, dan EX0 masing-masing adalah bit pembuat *enable port* serial, *timer* 1, INT1, *timer* 0, dan INT0.

Register IP terdiri atas bit-bit PS, PT1, PX1, PT0, PX0 berturut-turut adalah prioritas interupsi *port* serial, *timer* 1, INT1, *timer* 0, dan INT0. *Priority bit*=1 menandakan prioritas tinggi, dan *priority bit*=0 menandakan prioritas rendah.

2.5 Tomografi Konvensional[7]

2.5.1. Fungsi

Tomografi merupakan tipe spesial dari *imaging* yang dipakai untuk mendapatkan gambar diagnostik dari suatu lapisan khusus dari jaringan atau obyek yang tertutup oleh lapisan atau obyek lainnya. Ini diselesaikan dengan memanfaatkan peralatan tambahan yang memungkinkan tabung sinar-x dan film untuk bergerak dengan berporos pada titik *fulcrum* selama *ekspose*. Hasil radiografnya disebut tomogram, yang memperlihatkan sebuah gambar jelas dari sebuah obyek yang terletak di dalam bidang khusus dan mengaburkan obyek yang berada di atas dan di bawah bidang khusus tersebut.

2.5.2 Terminologi

Tomogram menunjukkan gambar dari bagian tubuh, film tipe ini kadang-kadang disebut *body section radiography*. Istilah lain yang sering digunakan untuk tomografi diantaranya planigrafi, stratigrafi, dan laminografi. International Commission on Radiological Units and Measures (ICRU) pada 1962 mengeluarkan istilah tomografi untuk semua bentuk *body section radiography*.

Beberapa istilah yang sering dipakai :

- *Tomogram* – radiograf yang dihasilkan oleh unit tomografi.
- *Fulcrum* – titik poros dari tongkat penghubung tabung sinar-x dan film.
- *Objective plane* (*focal plane*) – suatu bidang dimana obyek jelas dan fokus.

---

[7] Tortorici, M.,*Concept in Medical Radiographic Imaging* (Philadephia: W.B. Saunders Co., 1992) pp. 569-577.

- *Sectional thickness* - ketebalan dari obyek atau bidang fokus. (Ini merupakan variabel, dikontrol oleh sudut *ekspose* dan gerakan tabung)
- *Exposure angle* – sudut yang dihasilkan dari pergerakan berkas sinar-x.
- *Tube movement* (atau *shift*) – jarak perjalanan tabung.
- *Amplitude* – kecepatan pergerakan tabung diukur dalam inci atau cm/detik.
- *Tube trajectory* – konfigurasi geometrik atau bentuk pergerakan tabung.
- *Blur* - daerah yang kabur/rusak dari obyek di luar bidang obyektif.
- *Blur margin* - tepi luar dari obyek yang kabur.

2.5.3 *Tube Trajectories*

Ada 5 tipe dasar lintasan (*trajectory*) untuk pergerakan tabung dalam tomografi, dari yang sederhana sampai yang paling kompleks :

1. *Linear* ---------- (*unidirectional*)
2. *Elliptical*
3. *Circular*
4. *Spiral*
5. *Hypocycloidal*

} (*multi-directional*)

Pada lintasan linier (kadang-kadang tersedia dalam bentuk *vertilinear* dan *curvilinear*) tabung bergerak dalam satu arah, sehingga disebut sebagai "***unidirectional***". Lintasan tabung *elliptical*, *circular*, *spiral*, dan *hypocycloidal* semuanya bergerak pada beberapa arah, sehingga disebut sebagai ***multidirectional*** (*pluridirectional*).

2.5.4 Pergerakan Tabung *Unidirectional*

Tomografi linier atau *unidirectional* meliputi tipe peralatan yang tidak terlalu kompleks. Hal ini memanfaatkan meja sinar-x dasar dengan *bucky tray* dan tabung di atas dihubungkan dengan lengan penghubung metal atau batang. Batang ini melalui perlengkapan pengatur level *fulcrum*. Pelengkapan ini digunakan untuk mengatur ketinggian level *fulcrum*.

Pergerakan tabung dihasilkan oleh motor. Selama tabung bergerak sepanjang sumbu longitudinal meja, pengunci tabung longitudinal harus terbuka. Pengunci *bucky tray* dan pengunci sudut tabung juga harus terbuka untuk mengizinkan bergerak bebas.

2.5.5 Panel Kontrol

Unit tomografi dioperasikan oleh panel kontrol tersendiri. Pilihan pada panel kontrol bervariasi pada tiap unit. Ciri-ciri umum peralatan kontrol adalah untuk mengatur :

- kecepatan gerak tabung (dalam inci atau cm per detik) dengan amplitudo yang dapat diubah.
- *objective plane* (ketebalan bidang fokus, atau ketebalan setempat)
- arah atau tipe pergerakan tabung (pada peralatan dengan kemampuan *multidirectional*)
- menengahkan tabung
- level *fulcrum*

Beberapa unit dirancang sehingga semua fasilitas kecuali pengatur level *fulcrum* berada pada ruang kontrol. Pada peralatan ini, sudah umum bila pengatur level *fulcrum* ditempatkan langsung pada pengait fulcrum yang terhubung pada meja sinar-x. Peralatan lain mungkin juga mempunyai pemilih sudut *ekspose* yang ditempatkan pada meja sinar-x daripada di daerah pojok kontrol.

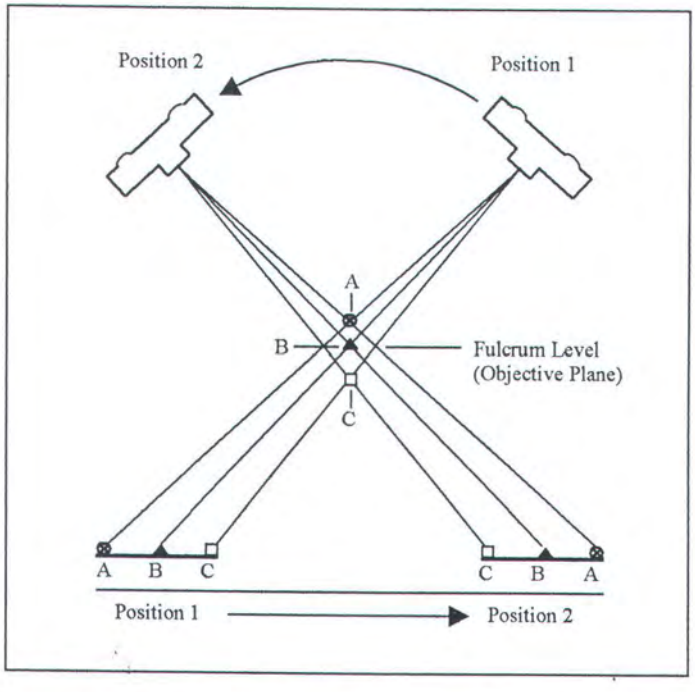
Sebagian besar dari obyek diparalel dengan pergerakan tabung pada peralatan dengan pergerakan tabung *unidirectional* atau linier. Hal ini membatasi jumlah pengaburan pada tomogram. Hal ini juga menciptakan gambar yang memiliki coretan. Kejernihan gambar mungkin juga diperbaiki dengan memaksimalkan jumlah dari pergerakan tegak lurus dari tabung terhadap obyek. Hal ini dapat diterima dengan mengubah tipe pergerakan tabung menjadi *multidirectional*.

2.5.6 Pergerakan Tabung *Multidirectional*

Dari empat tipe *multidirectional*, yang paling sederhana adalah gerakan ellips, yang memiliki sedikit variasi dari linier. Gerakan sirkular satu tahap lebih kompleks daripada ellips. Dua gerakan *multidirectional* yang paling kompleks adalah spiral dan *hypocycloidal*. Semakin kompleks gerakan *multidirectional* semakin tipis bidang obyektif yang dihasilkan dan semakin jelas gambar yang dihasilkan.

2.5.7 Fulcrum

Fulcrum adalah titik poros dimana tabung sinar-x dan film bergerak.Titik poros ini sangat penting karena semua struktur yang berada pada bidang ini (bidang obyektif) dan diparalel dengan pergerakan tabung tetap tajam dan fokus karena semuanya pada posisi yang sama (struktur tidak bergerak) pada film selama *ekspose*. Sebaliknya, semua obyek yang berlokasi di luar bidang obyektif, baik di atas maupun di bawah diproyeksikan dari satu titik pada film ke titik lainnya. Sebagai contoh, lihat titik A pada gambar 2.15, ketika film dan tabung bergerak dari posisi 1 ke posisi 2. Titik A semula ada pada tepi kiri film pada posisi 1, tetapi akhirnya berada pada tepi kanan pada posisi 2 menghasilkan gerakan atau pengaburan obyek pada titik A.



Gambar 2.15 Prinsip "Bluring" Tomografi

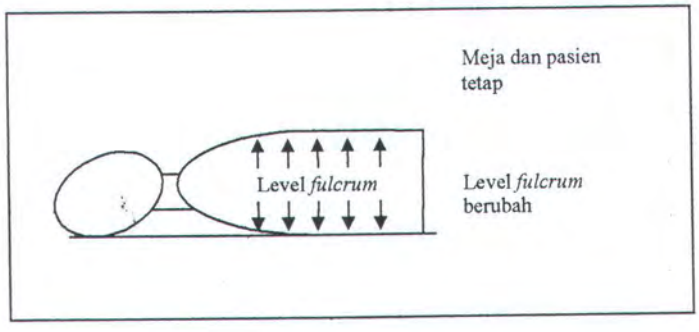
Titik C di bawah level *fulcrum*, dengan cara yang sama dikaburkan ketika diproyeksikan dari satu tepi film sampai tepi film lainnya. (Titik C berada pada tepi kiri film pada posisi 1, dan akhirnya berada pada tepi kiri film pada posisi 2).

Jumlah gerakan dari struktur-struktur ini ditentukan oleh jarak obyek dari *fulcrum*. Konsekuensinya, obyek yang tetap tidak bergerak terlihat jelas (tajam) pada tomogram dan obyek di atas dan di bawah *fulcrum* bergerak sehingga hasilnya kabur. Hal ini didasarkan pada prinsip pengaburan tomografi (*tomographic blurring principle*).

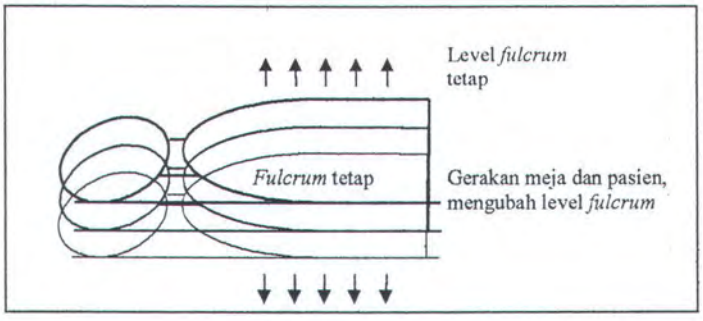
2.5.8 *Fulcrum* Variabel dan *Fulcrum* Tetap

Level *fulcrum* digunakan untuk menentukan level anatomi yang diinginkan untuk diambil gambarnya. Hal ini bisa dicapai dengan dua cara. Metode yang paling sederhana adalah menggunakan *fulcrum* variabel (dapat diatur atau bergerak), ditunjukkan pada gambar 2.16.

Metode kedua meliputi *fulcrum* tetap dan tinggi meja yang dapat diatur (gambar 2.17). Cara ini lebih sering dipergunakan pada peralatan tomografi *multidirectional* atau peralatan khusus. Pada sistem ini, level *fulcrum* dapat diubah dengan menggerakkan pasien dan meja ke atas atau ke bawah sampai bidang yang diinginkan pada pasien berada pada level *fulcrum* tetap.



Gambar 2.16 *Fulcrum* variabel



Gambar 2.17 *Fulcrum* tetap

2.5.9 Menentukan Level Fokus dan Pemusatan

Untuk menentukan pemusatan dan menentukan di mana level *fulcrum* seharusnya. Dua buah radiograf konvensional 90° diambil, misalnya PA atau AP dan dada *lateral* untuk tomogram dari daerah yang diduga di dalam paru-paru. Satu radiograf, PA, menunjukkan lokasi obyek yang diinginkan relatif terhadap posisi *lateral* (kiri/kanan), misal 5 cm ke kanan dari tulang belakang. Radiograf *lateral* digunakan untuk menentukan posisi *anterior/posterior*, misalnya 10 cm *posterior* ke arah tulang dada. Kedua radiograf ini berguna untuk menentukan lokasi *superior/inferior* dari obyek.

Dengan menggunakan radiograf PA dan *lateral*, dapat diketahui dengan pasti lokasi dasar dari organ, teknolog dapat memperkirakan sisi dari obyek atau area khusus yang diinginkan. Tomogram pemandu awal diambil dengan *fulcrum* dipasang pada level yang diperkirakan dari area khusus yang diinginkan. Pemusatan akan dikerjakan dengan menentukan jarak tertentu sesuai dengan perhitungan dari radiograf PA dan/atau *lateral* dari peta posisi yang telah diketahui.

2.5.10 Pengaburan *(Blur)*

Pengaburan didefinisikan sebagai area yang rusak dari obyek di luar bidang obyektif. Dalam tomografi, suatu struktur yang menumpuki obyek yang diinginkan dikaburkan. Obyek atau struktur yang kabur pada pasien ini adalah yang berada di atas maupun di bawah level yang diinginkan pada level *fulcrum*.

Ada empat faktor pengontrol dan faktor yang mempengaruhi jumlah pengaburan, yaitu :

(1) Jarak obyek dari bidang obyektif

(2) Sudut *ekspose* (besarnya sudut pergerakan tabung dari posisi 1 ke posisi 2).

(3) Jarak obyek dari film

(4) Pergerakan tabung (bentuk gerakan tabung).

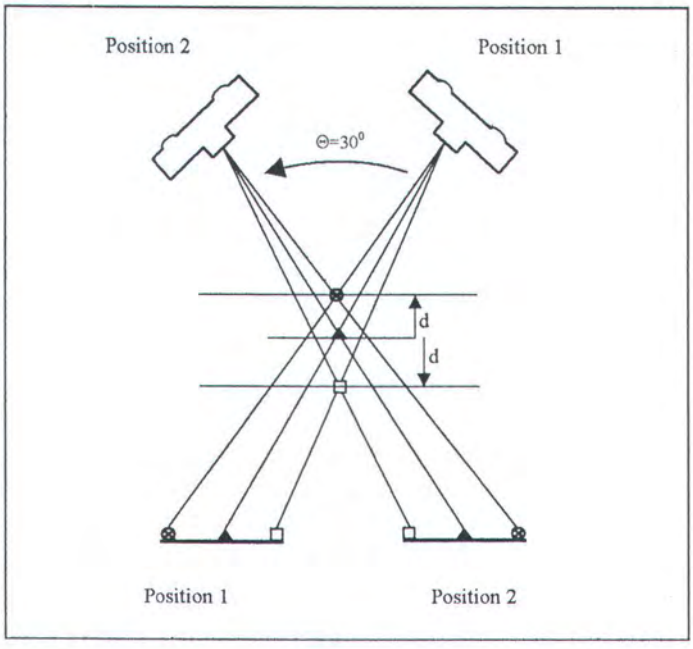
Besarnya pengaburan dipengaruhi oleh dua faktor pertama, (1) jarak obyek dari bidang obyektif dan (2) sudut *ekspose*, ditunjukkan dengan rumus berikut :

$$Movement = 2d \tan \Theta/2 \qquad (2.14)$$

Movement adalah ukuran pengaburan dimana d adalah jarak obyek dari *fulcrum*, dan Θ adalah sudut *ekspose*.

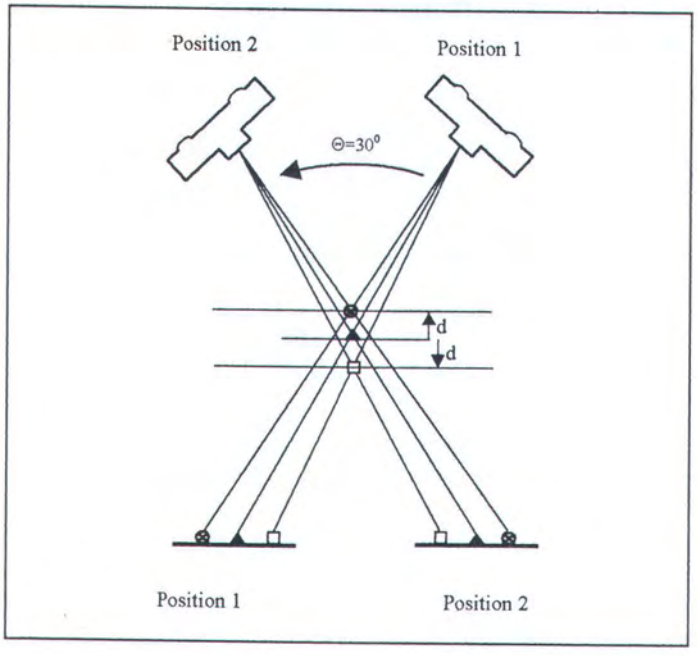
(1) Jarak obyek dari bidang obyektif (d) : Rumus di atas menyatakan bahwa jika semua faktor konstan, kecuali d, maka apabila d bertambah, *movement* atau pengaburan bertambah. Hal ini ditunjukkan dengan membandingkan gambar 2.18 dan 2.19 dimana sudut *ekspose* konstan pada 30°, tetapi d berubah (jarak dari bidang *fulcrum* dengan obyek di atas maupun di bawah). Gambar 2.18, dengan jarak d lebih besar, memiliki *movement* yang lebih besar dari obyek A dan C pada film dari posisi 1 ke posisi 2, sehingga menambah pengaburan.

Hal ini menunjukkan bahwa obyek pada tubuh yang lebih jauh jaraknya dari bidang fokus memiliki *movement* yang lebih besar sehingga menambah pengaburan.



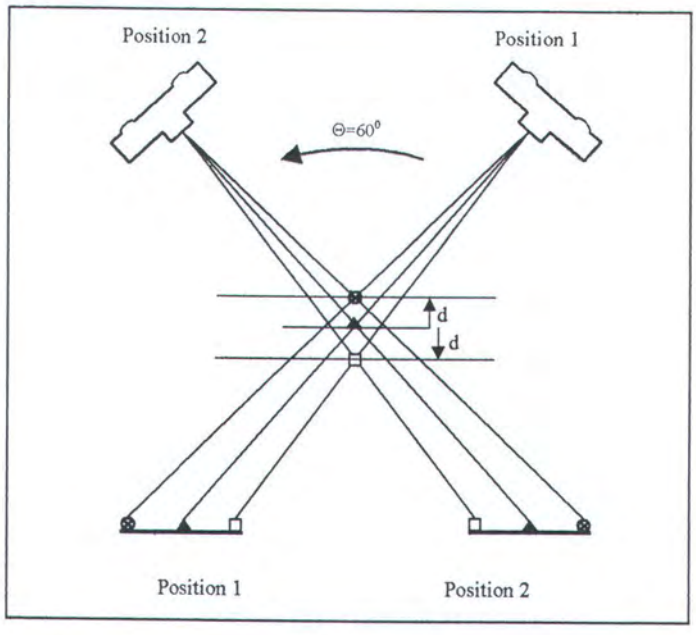
Gambar 2.18 Penambahan d, berarti penambahan *movement/blurring*

(2) Sudut *ekspose* (Θ) : Rumus tersebut di atas juga menunjukkan bahwa apabila hanya Θ (sudut *ekspose*) yang bertambah dan faktor lainnya konstan, maka *movement* atau pengaburan bertambah. Hal ini ditunjukkan dengan membandingkan gambar 2.20 pada sudut *ekspose* 60°, dengan sudut 30° pada gambar 2.19. Sudut *ekspose* 60° menambah *movement* dari obyek A dan C meskipun d tidak berubah, sehingga menambah pengaburan.

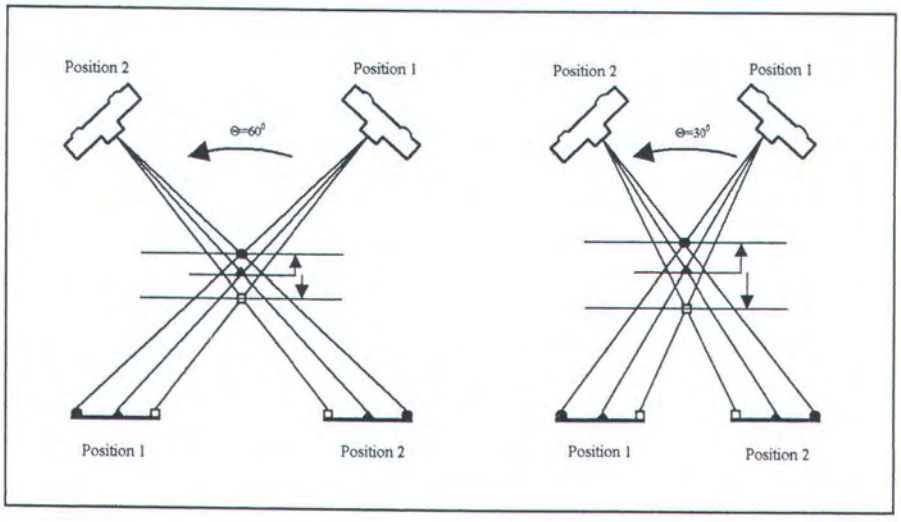


Gambar 2.19 Pengurangan d, berarti pengurangan *movement/blurring*

Kesimpulan rumus : Rumus di atas menunjukkan bahwa jika jarak obyek dari bidang obyektif bertambah dan/atau jika sudut *ekspose* bertambah, terjadi penambahan jumlah pengaburan. Efek samping dari penambahan pengaburan ini adalah menipisnya bidang fokus (bidang obyektif) seperti yang ditunjukkan pada pembahasan selanjutnya (gambar 2.21).



Gambar 2.20 Penambahan sudut *ekspose*, berarti penambahan *movement* dan menambah pengaburan



Gambar 2.21 *Sectional Thickness* (ketebalan bidang obyektif)

(3) Jarak obyek dengan film : Faktor ketiga yang mempengaruhi pengaburan adalah jarak obyek dari film. Jika jarak dari film bertambah, pengaburan bertambah. Ini bukan merupakan variabel kontrol yang dapat diatur, tetapi

ditentukan oleh ketebalan tubuh, atau oleh lokasi umum atau jarak obyek yang akan diambil gambarnya terhadap film.

(4) Bentuk pergerakan tabung : Faktor keempat dan terakhir yang mempengaruhi atau mengontrol jumlah pengaburan adalah bentuk pergerakan tabung. Pengaburan maksimum dari obyek terjadi bila struktur tegak lurus terhadap arah pergerakan tabung. Pada pergerakan *unidirectional* atau linier, pergerakan tabung hanya satu arah. Hal ini menghasilkan obyek yang paralel dengan pergerakan tabung dalam porsi yang besar. Sehingga lebih sedikit pengaburan terjadi dengan pergerakan tabung *unidirectional*.

Sebaliknya, pergerakan tabung *multidirectional* menghasilkan sedikit obyek yang paralel dengan total pergerakan tabung. Sehingga pergerakan tabung ellips atau sirkular menghasilkan pengaburan yang lebih besar.

Bentuk pengaburan maksimum : Bentuk pergerakan tabung spiral dan *hypocycloidal* menghasilkan pengaburan maksimum di dalamnya termasuk dimensi vertikal sebagai bagian dari pergerakan *multi-tiered* yang menghasilkan pengaburan maksimum dari struktur yang dekat maupun jauh dari bidang fokus. Pergerakan ini paling bermanfaat pada prosedur tomografi yang meliputi tengkorak seperti struktur bagian dalam mata yang kecil yang memerlukan bidang fokus 1 mm atau lebih kecil.

2.5.11 *Sectional Thickness* (ketebalan bidang obyektif)

Pengaburan bertambah jika jarak dari level *fulcrum* atau bidang obyektif bertambah. Sehingga hal ini menghasilkan proses bertahap dari pengaburan dengan struktur yang dekat dengan level *fulcrum* sedikit kabur, dan struktur yang

paling jauh paling kabur. Demikian juga, mata manusia dibatasi dalam kemampuannya untuk membedakan antara kabur dan tidak kabur. Jadi mata manusia menerima besar pengaburan tertentu. Jumlah pengaburan yang dapat ditangkap mata ini subyektif dan bervariasi pada setiap individu. Faktor kombinasi ini menghasilkan apa yang disebut dengan *sectional thickness* atau ketebalan bidang obyektif.

Semakin besar pengaburan terjadi, semakin tipis bidang obyektif. Faktor utama yang mempengaruhi *sectional thickness*, dan yang dapat dikontrol operator adalah sudut *ekspose* (gambar 2.21). Ini bermanfaat untuk mengatur ketebalan bidang obyektif dan obyek yang akan diambil gambarnya. Obyek-obyek kecil lebih baik diambil gambarnya menggunakan bidang obyektif yang tipis dengan sudut *ekspose* lebih besar. Dan untuk obyek yang besar, seperti paru-paru, harus menggunakan bidang obyektif yang tebal dengan sudut *ekspose* yang kecil.

Ada beberapa macam tomografi konvensional , yaitu :

- *Autotomography (Breathing Technique)*
- *Pantomography (Panorex)*
- *Skip Tomography*
- *Computed Tomography (CT).*

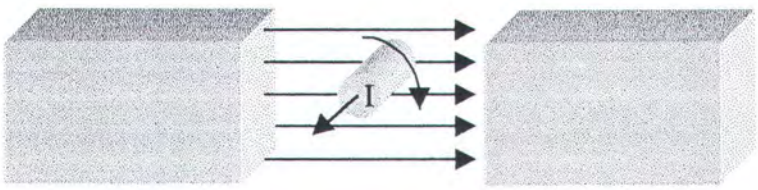
2.6 Motor Arus Searah (DC)

Motor arus searah merupakan suatu alat yang berfungsi mengubah tenaga listrik arus searah menjadi tenaga gerak atau tenaga mekanik yang berupa putaran pada rotor. Motor arus searah mempunyai prinsip kerja berdasarkan percobaan

Lorentz[7] yang menyatakan "Jika sebatang penghantar listrik yang berarus berada di dalam medan magnet maka pada kawat penghantar tersebut akan terbentuk suatu gaya". Gaya yang terbentuk sering dinamakan Gaya Lorentz. Untuk menentukan arah gaya dapat digunakan kaidah tangan kiri dan besarnya adalah :

$$F = I\,(L \times B) \quad \text{Newton} \tag{2.15}$$

$$= I \,.\, L \,.\, B \sin\theta \quad \text{Newton} \tag{2.16}$$



Gambar 2.22 Prinsip kerja motor dc

Persamaan dari motor dc adalah sebagai berikut :

Gambar 2.23 Persamaan motor dc

$$Va = Ia\,Ra + Ea \tag{2.17}$$

$$Ea = N \,.\, \phi \,.\, C \tag{2.18}$$

Dimana : Ia = arus jangkar

Ra = tahanan jangkar

---

[7] _____, *Teknologi Motor dan Generator* (_____, 1988), p.85

Ea = GGL lawan jangkar

N = kecepatan putar (rps)

C = konstanta

ϕ = *fluks*

Besarnya daya mekanik adalah :

$$Pm = V \cdot Ia - Ia^2 \cdot Ra \qquad (2.19)$$

Untuk kondisi daya mekanik maksimum ialah :

$$\frac{Pm}{dIa} = 0, \qquad (2.20)$$

$$\text{maka} \quad Ea = Ia \cdot Ra = \frac{V}{2} \qquad (2.21)$$

Dari persamaan tegangan motor maka didapat kecepatan motor dc adalah :

$$N = \frac{V - Ia.Ra}{\phi.C} \text{ rps} \qquad (2.22)$$

Torsi yang dihasilkan oleh motor dc sebanding dengan kuat medan dan arus jangkar.

$$T = K \cdot \phi \cdot Ia \quad Nm \qquad (2.23)$$

Dimana : T = torsi

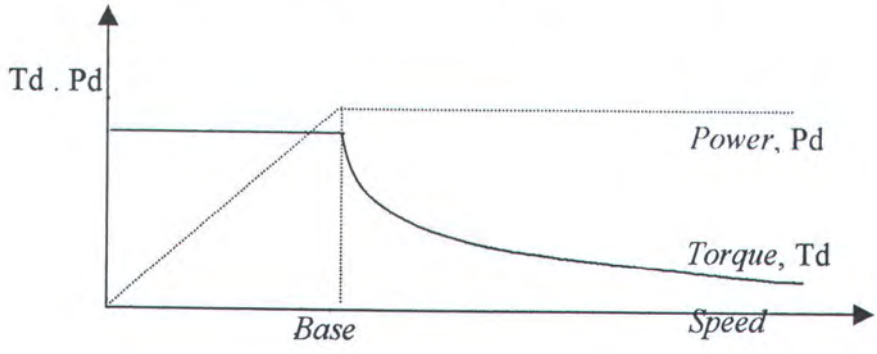
K = konstanta perancangan

2.6.1. Pengatur Kecepatan dan Karakteristik Daya Motor DC

Kecepatan putaran motor DC secara umum dapat diatur dengan mengubah besarnya tegangan jangkar, arus medan, dan torsi. Perkalian antara kecepatan dan torsi menghasilkan daya motor.

Selama motor belum mencapai maksimum, maka kecepatan dapat dinaikkan dengan asumsi torsi konstan. Saat daya mencapai maksimum, yaitu pada saat kecepatan mencapai "*base speed*", maka hasil kali kecepatan dan torsi adalah konstan, akibatnya jika kecepatan ditambah maka torsi akan menurun. Karena itu besarnya kecepatan maksimum motor dc tergantung pada daya motor dan torsi.

Karakteristik daya motor ditunjukkan gambar berikut :



Gambar 2.24 Karakteristik daya motor dc

Untuk mengontrol tegangan yang diberikan ke motor dc digunakan metode PWM (*pulse width modulation*). Ada dua metode untuk mengendalikan kecepatan putaran motor, yaitu :

- Mengoperasikan motor pada frekuensi yang tetap, yang diubah adalah besarnya periode waktu *on* (*pulse width modulation*).
- Mengoperasikan motor pada frekuensi yang berubah, periode waktu *on* dibuat konstan, sedangkan frekuensi dibuat bervariasi (*Frequency Modulation*).

Untuk mengendalikan arah dan kecepatan putaran motor menggunakan PWM, berdasarkan sinyal kontrol yang digunakan dibedakan menjadi dua macam[8] :

- *Simple, locked anti-phase PWM*, cara ini menggunakan sebuah sinyal kontrol PWM yang memberikan informasi arah sekaligus besarnya kecepatan. *Duty-cycle* sinyal PWM 50% menunjukkan arah dan kecepatan nol.
- *Sign/magnitude PWM*, terdiri atas dua buah sinyal terpisah yaitu sinyal arah putaran dan sinyal besarnya kecepatan motor.

2.7 Sensor

Sensor berfungsi untuk mengukur parameter yang diperlukan pada sistem kontrol. Dalam tugas akhir ini parameter yang diukur adalah posisi benda yang akan diatur. Untuk mendeteksi posisi ini digunakan resistor variabel (VR). Perubahan posisi benda mengakibatkan perubahan resistansi VR yang selanjutnya dikonversikan menjadi perubahan tegangan DC.

2.8 Analog to Digital Converter

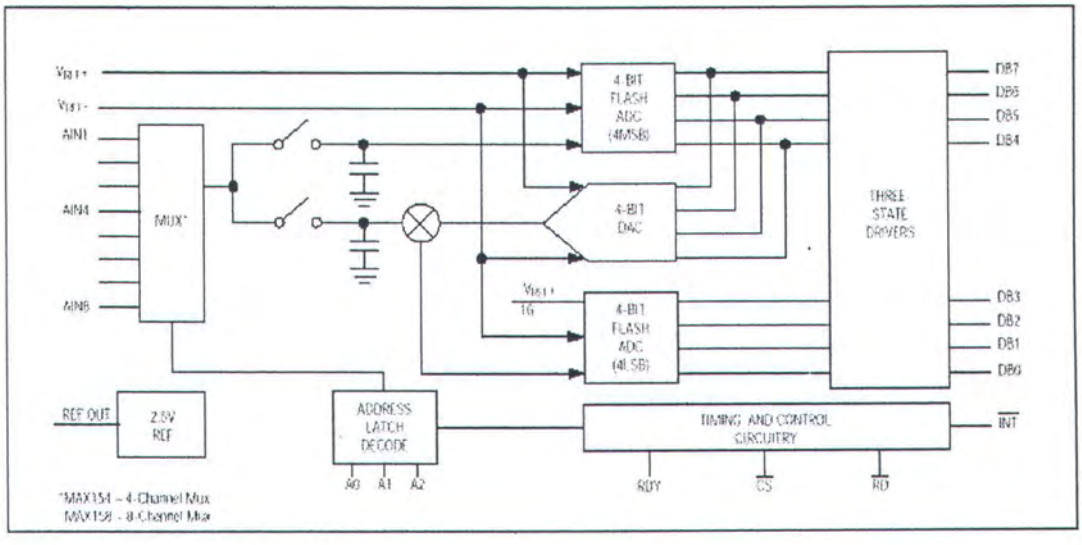
ADC yang digunakan adalah MAX 154 yang merupakan *analog-to-digital converter* (ADC) multi kanal kecepatan tinggi. MAX 154 mempunyai empat kanal *input* analog. Waktu konversi yang diperlukannya adalah 2.5 μs. MAX 154 juga memiliki tegangan referensi 2.5 volt internal, sehingga membentuk sistem akuisisi data kecepatan tinggi yang lengkap. ADC ini di dalamnya sudah ada *track/hold*, mengurangi keperluan penggunaan eksternal *track/hold*. Variasi *input*

---

[8] _____, *National Power IC Databook* (Crawfordsville : National Semiconductor, 1995), p.4-47

berkisar antara 0V sampai dengan +5V, meskipun beroperasi pada tegangan suplai tunggal +5V.

MAX 154 menggunakan teknik konversi "*half-flash*" (gambar 2.14). Dua buah konverter *flash* ADC 4-bit digunakan untuk menghasilkan keluaran 8 bit. Menggunakan 15 komparator, *flash* ADC 4-bit MS (*most significant*) membandingkan tegangan *input* yang tidak diketahui dengan *ladder* referensi dan menyediakan 4 data bit *upper*. DAC internal menggunakan MS bit untuk membangkitkan sinyal analog dari *flash* konversi pertama. Tegangan sisa menunjukkan perbedaan antara tegangan input yang tidak dikenal dengan tegangan DAC yang kemudian dibandingkan dengan *ladder* referensi oleh 15 *flash* komparator LS (*least significant*) untuk menghasilkan 4 data bit *lower*.



Gambar 2.25 Diagram fungsi IC MAX 154

BAB III

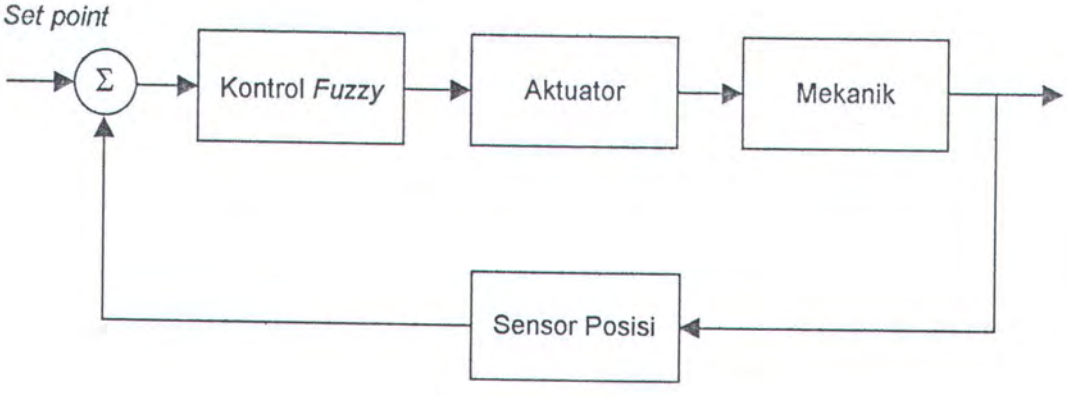
PERANCANGAN ALAT

Pada bab ini akan dibahas perancangan sistem secara keseluruhan yang meliputi perancangan mekanik, perancangan perangkat keras, dan perancangan perangkat lunak. Langkah-langkah yang dilakukan adalah :

- merancang diagram blok sistem secara keseluruhan dan cara kerja sistem secara umum
- merancang mekanik sesuai dengan sistem yang direncanakan
- merancang rangkaian elektronik yang dibutuhkan pada setiap blok sistem
- merancang perangkat lunak untuk menunjang kerja sistem

3.1 Perancangan System

Diagram blok kontrol gerakan tabung tomograf yang akan dibuat ditunjukkan pada gambar 3.1.



Gambar 3.1 Diagram blok kontrol gerakan tabung tomograf

Pada diagram blok di atas, sistem yang direncanakan terdiri atas empat blok, yaitu blok kontrol *fuzzy* berupa mikrokontroler AT89C51, blok aktuator berupa *driver* kecepatan dan arah gerak motor, blok mekanik, dan blok sensor posisi berupa resistor variabel. Untuk masukan *set point* digunakan tombol kontrol.

Cara kerja kontrol gerakan tabung tomograf ini adalah motor akan menggerakkan tabung tomograf ke posisi yang sesuai dengan *set point*, posisi ini merupakan posisi awal *ekspose* yang akan dilakukan yang terletak pada sebelah kanan posisi normal/tegak dengan besar sudut tertentu yang dibentuk lengan penyangga dengan garis vertikal. Setelah tombol *ekspose* diaktifkan, motor akan menggerakkan tabung tomograf ke posisi yang berlawanan (ke arah kiri) dengan besar sudut yang sesuai dengan *set point*. Setelah itu motor menggerakkan tabung ke posisi normal kembali (posisi tegak).

Kecepatan motor diatur sedemikian rupa sehingga diperoleh pencapaian posisi secara cepat dan tepat. Untuk mengatur kecepatan motor ini dipergunakan kontrol *fuzzy* dengan input berupa selisih posisi yang dicapai. Apabila selisih posisi besar, maka kecepatan motor besar, bila selisih posisi kecil, kecepatan motor kecil, dan bila selisih posisi nol, maka motor berhenti berputar.

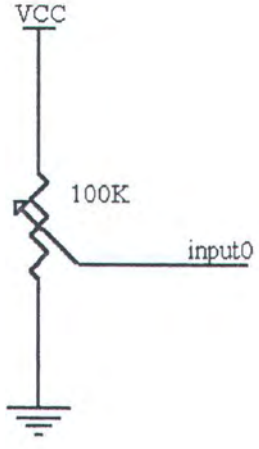
3.2 Perancangan Mekanik

Perancangan mekanik yang dilakukan adalah untuk mendukung perancangan kontrol *fuzzy*, yaitu sebagai alat simulasi. Mekanik dirancang mendekati bentuk aslinya pada peralatan tomograf, namun hanya sebagian saja,

yaitu bagian penggerak tabung sinar-x. Mekanik yang dirancang adalah penggerak tabung dengan jenis gerakan tabung *unidirectional*.

Penggerak yang digunakan adalah motor arus searah. Motor dihubungkan dengan rantai sepanjang lintasan gerak tabung. Benda yang digerakkan dikaitkan pada rantai. Untuk memperkuat torsi motor digunakan roda gigi dengan perbandingan 3 : 47.

Sensor posisi berupa resistor variabel *multitune* yang pada kedua terminalnya diberikan tegangan listrik sebesar 5 Volt. Keluaran sensor posisi berupa perubahan tegangan sesuai dengan posisi benda.



Gambar 3.2 Sensor sudut posisi

3.3 Perancangan Perangkat Keras

3.3.1 Mikrokontroler AT89C51

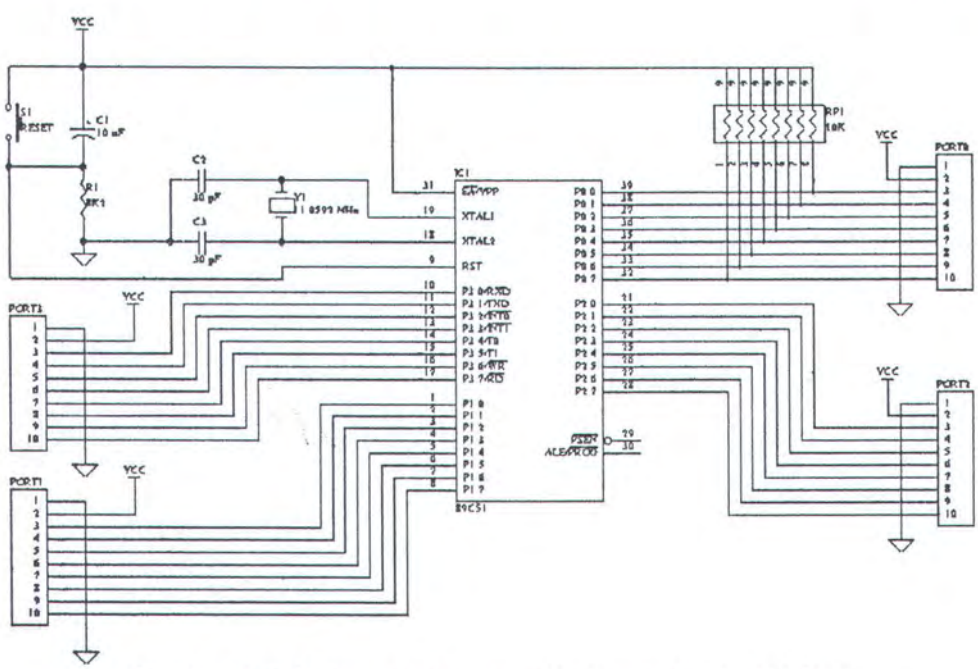
Sistem minimum dengan basis mikrokontroler AT89C51 yang dibuat dimanfaatkan sebagai kontrol *fuzzy* sekaligus pengendali sistem secara

keseluruhan. Data yang diolah keluar masuk melalui *port-port* yang ada, baik dalam bentuk data 8-bit maupun data dalam 1 bit yang berdiri sendiri.

Mikrokontroler AT89C51 sudah memiliki internal flash ROM sebesar 4 K *byte* yang bisa diisi program. Sistem minimum yang dibuat hanya memerlukan komponen tambahan berupa *clock* 11.0592 MHz dan rangkaian *autoreset*.

Port 0 digunakan untuk menyalakan LED sebagai indikator arah gerakan dan besar sudut. Untuk keperluan ini digunakan P0.0 sampai dengan P0.4. Kaki P0.5 tidak digunakan. Kaki P0.6 dan P0.7 digunakan sebagai keluaran sinyal PWM dan sinyal arah gerakan motor.

Port 1 digunakan untuk memasukan kontrol *set point* yang diinginkan, kontrol arah gerakan yang diinginkan, dan kontrol penekanan *ekspose*. *Port* ini dihubungkan dengan bagian tombol kontrol. *Port* 2 digunakan untuk membaca data digital hasil konversi ADC. *Port* 3 hanya digunakan 2 kaki saja, yaitu P3.0 untuk kontrol ADC dan P3.2 sebagai masukan interupsi eksternal (sebagai INT0).



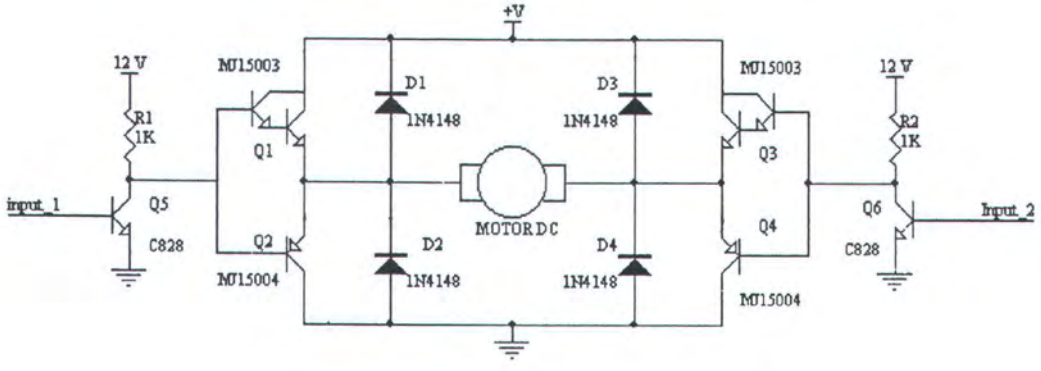
Gambar 3.3 Sistem minimum mikrokontroler AT89C51

3.3.2 *Driver* Kecepatan dan Arah Putaran Motor

Untuk mendapatkan pengaturan kecepatan motor dc yang arah putarannya dapat dibalik, serta hanya menggunakan sebuah tegangan sumber, rangkaian penggerak yang dipakai menggunakan empat buah transistor dengan konfogurasi jembatan-H, dimana masing-masing pasangan transistor PNP dan NPN dikonfigurasikan *push-pull*. Cara kerja *driver* adalah sebagai berikut :

1. Saat Q1 dan Q4 bersama-sama saturasi (*on*), tegangan sumber tampak melintang pada teminal motor dan arus jangkar akan mengalir, maka motor akan berputar searah jarum jam (*cw*)
2. Pada saat motor masih berputar *cw*, saat Q1 dan Q4 *cut off*, bila Q3 lebih dulu saturasi dari Q2 mengakibatkan motor yang bekerja sebagai generator mengembalikan dayanya melalui Q3 dan D1. Sedangkan bila Q2 lebih dulu saturasi dari Q3 mengakibatkan motor yang bekerja sebagai generator mengembalikan dayanya melalui Q2 dan D4.
3. Saat Q2 dan Q3 bersama-sama saturasi (*on*), tegangan sumber tampak melintang pada teminal motor dan arus jangkar akan mengalir, maka motor akan berputar berlawanan arah dengan jarum jam (*ccw*)
4. Pada saat motor masih berputar *ccw*, saat Q2 dan Q3 *cut off*, bila Q1 lebih dulu saturasi dari Q4 mengakibatkan motor yang bekerja sebagai generator mengembalikan dayanya melalui Q1 dan D3. Sedangkan bila Q4 lebih dulu saturasi dari Q1 mengakibatkan motor yang bekerja sebagai generator mengembalikan dayanya melalui Q4 dan D2.

Masukan berupa sinyal PWM masuk pada *input1*, dan sinyal *high* atau *low* untuk menentukan arah putaran motor masuk pada *input2*. Sinyal ini dihasilkan oleh mikrokontroler dan dikeluarkan melalui *port* 0 pada kaki P0.6 dan P0.7. Sinyal PWM yang dikeluarkan sesuai dengan masukan data yang berasal dari keluaran dari hasil pengolahan kontrol *fuzzy* .



Gambar 3.4 Driver motor konfigurasi jembatan H

3.3.3 *Analog to Digital Converter* MAX154

Rangkaian ini digunakan sebagai pengubah sinyal masukan analog dari sensor posisi menjadi data digital 8-bit. Selanjutnya data tersebut dikirim ke mikrokontroler dan disimpan dulu di *register* sebelum diolah. Pada tugas akhir ini hanya diperlukan pengubahan satu sinyal analog saja, sehingga hanya menggunakan satu kanal dari 4 kanal yang tersedia, yaitu kanal 1 (AIN1).

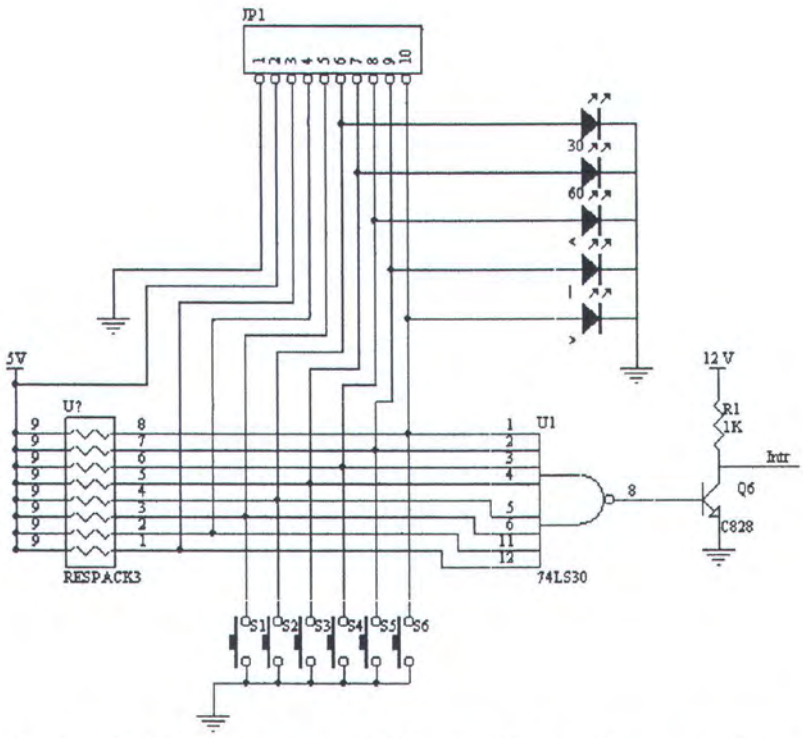
Pada ADC ini, VREF- diberi tegangan 0 Volt dan VREF+ diberi tegangan 5 Volt yang merupakan tegangan pada terminal sensor posisi. Masukan dari sensor posisi berkisar antara VREF- dan VREF+. Jadi kisaran data digital yang dihasilkan ADC dari 0000000 sampai dengan 11111111.

Kaki alamat A0 dan A1 langsung diisi logika 0 dengan menghubungkan ke *ground* secara langsung. Kaki CS (*Chip Select*) yang aktif low juga di-*ground*-kan, sehingga ADC bekerja terus. Dan untuk pengambilan data dikontrol oleh mikrokontroler melalui *port* 3 (P3.0) untuk mengontrol kaki RD (*Read*) pada ADC. Data diambil oleh mikrokontroler melalui *port* 2.

3.3.4 Tombol Kontrol

Tombol kontrol digunakan untuk mengatur kerja tomograf terdiri atas enam buah saklar *push-on*. Tombol ini digunakan untuk mengatur pergeseran tabung menuju posisi awal *ekspose* baik ke kanan (S6) maupun ke kiri (S4), untuk mengembalikan posisi tabung tomogaf ke tengah (S5), untuk menentukan besar sudut *ekspose* (S2 dan S3), dan sebuah tombol untuk mewakili tombol *ekspose* (S1).

Tombol-tombol ini terhubung ke *port* 1. Selama tidak ada penekanan, kaki-kaki *port* 1 berlogika 1. Bila tombol ditekan, kaki *port* 1 yang terhubung ke tombol tersebut berlogika 0. Selain terhubung ke *port* 1 tombol kontrol tersebut menjadi masukan IC TTL NAND 8 masukan, 74LS30. Keluaran 74LS30 ini di-*invert*-kan, sehingga fungsi gabungan yang terbentuk adalan fungsi AND 8 masukan. Selama tidak ada penekanan, sinyal high dihasilkan. Bila ada penekanan tombol, maka akan menghasilkan sinyal *low* yang selanjutnya mengaktifkan *interrupt 0* mikrokontroler. Jadi terjadi interupsi bila ada penekanan tombol.



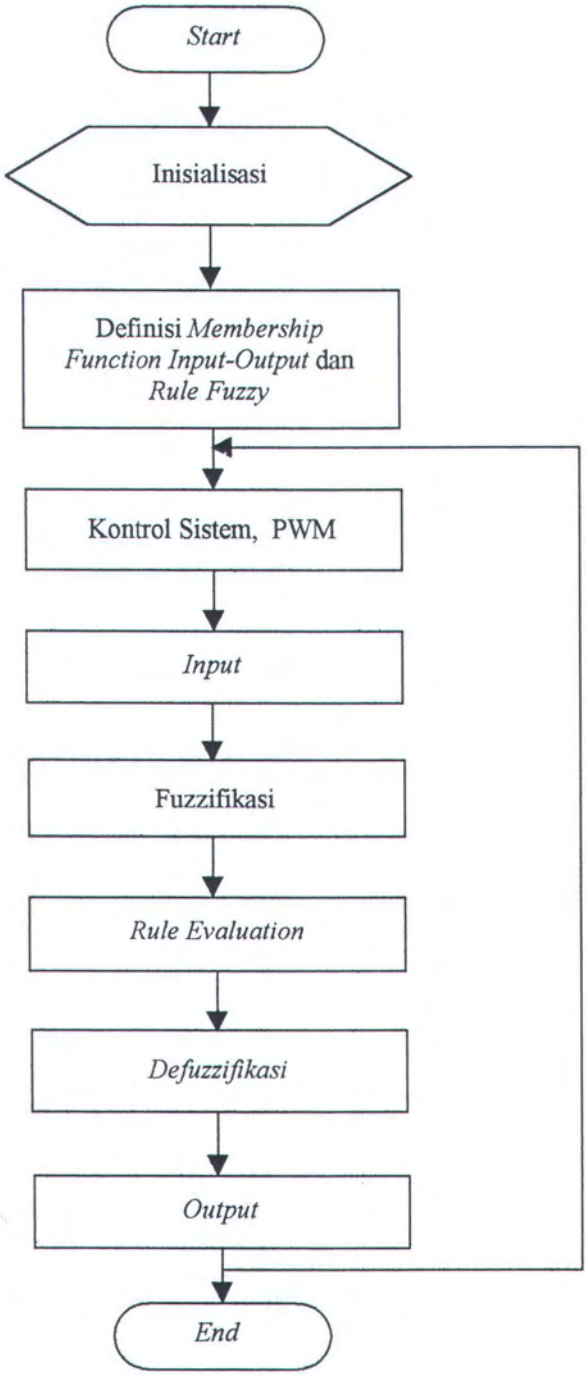
Gambar 3.5 Rangkaian tombol kontrol dan indikator gerakan

3.4 Perancangan Perangkat Lunak

Dalam tugas akhir ini perangkat lunak digunakan untuk pemrograman mikrokontroler agar berfungsi sebagai kontroler *fuzzy*. Di samping itu juga dimanfaatkan untuk membangkitkan sinyal PWM yang digunakan untuk mengatur kecepatan dan arah gerakan motor. Mikrokontroler juga mengatur kerja dari ADC dan kontrol sistem keseluruhan.

Masukan dari tombol kontrol berupa *set point* posisi yang diinginkan masuk melalui *port* 1 dan masukan dari ADC berupa posisi yang telah dicapai masuk melalui *port* 2. Dari dua masukan ini diperoleh data posisi yang diinginkan dan dapat dihitung kesalahan posisi dan perubahan kesalahan posisi. Data ini digunakan sebagai masukan kontrol *fuzzy* untuk diolah dan memberikan respon berupa sinyal kontrol arah dan kecepatan motor.

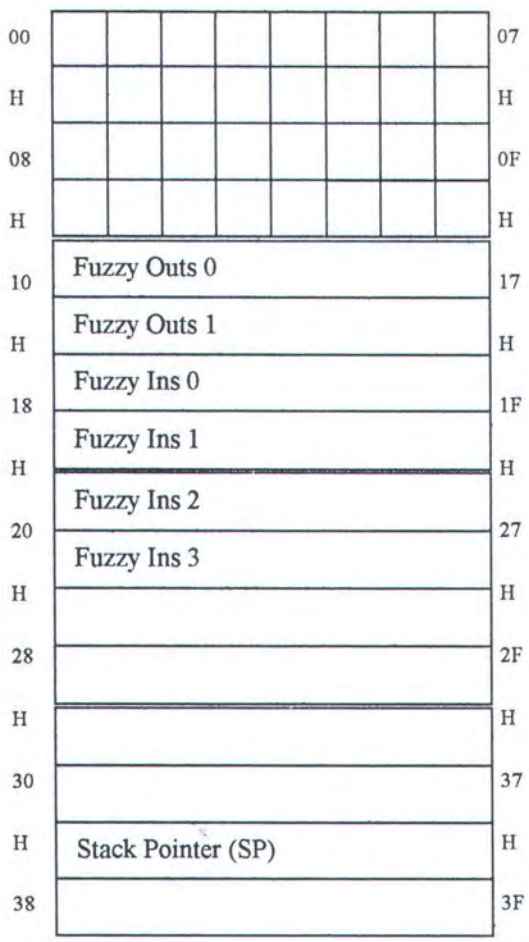
Program mikrokontroler yang dibuat memiliki beberapa bagian atau fungsi seperti digambarkan pada *flowchart* di bawah ini.



Gambar 3.6 *Flowchart* program mikrokontroler

Sinyal kontrol yang dihasilkan berupa sinyal satu bit sebagai penunjuk arah putaran motor dan sebuah sinyal dalam bentuk PWM. Keluaran sinyal kontrol dikeluarkan melalui kaki P0.6 dan P0.7. Untuk menghasilkan sinyal kontrol ini berdasarkan sebuah keluaran dari kontrol *fuzzy* berupa data 8-bit. Apabila data yang akan dikonversi kurang dari 80H, maka keluaran P0.7 *high*. Dan apabila data yang akan dikonversi lebih dari atau sama dengan 80H, maka P0.7 *low*. Sinyal keluaran pada P0.7 ini digunakan sebagai penentu arah putaran motor. Sedangkan sinyal PWM dikeluarkan melalui P0.6.

3.4.1 Mikrokontroler AT89C51 sebagai Kontroler *Fuzzy*



00H-03H : variabel inferensi fuzzy
04H-05H : kosong
06H-07H : untuk perhitungan DError
08H-09H : data panjang siklus program
0AH : masukan PWM
0BH : perhitungan normalisasi output
0CH : posisi saat ini, pembacaan dari P2
0DH : *latch* penekanan tombol kontrol
0EH : data penekanan tombol
0FH : data *set point*
10H : COGDEX
11H : LP_COUNT
12H : SUMDEX
13H : kosong
14H : perhitungan defuzifikasi
15H-17H : SUM_OF_PROD
18H-1BH : CURRENT_INS
1CH-1DH : COG_OUTS
1EH-!FH : SUM_OF_FUZ
20H-2FH : FUZ_OUTS
30H-4FH : FUZ_INS
50H-6FH : kosong

Gambar 3.7 Penggunaan RAM pada AT89C51

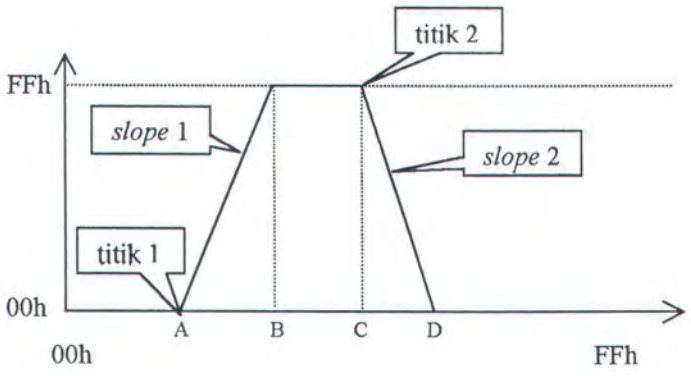
Program kontrol *fuzzy* yang dibuat memiliki maksimum empat variabel *input membership function* dan dua variabel *output membership function*. Masukan dan keluaran kontroler masuk dan keluar melalui *port* 2, dalam bentuk data digital 8-bit. Sebelum diproses, data *input-output* ini disimpan di *register* pada alamat 18H sampai dengan 1DH. *Port* 0 difungsikan sebagai keluaran indikator arah dan besar sudut gerakan dan juga sebagai keluaran sinyal PWM. Pada bagian *fuzzy*, setiap variabel masukan dan keluaran memiliki maksimum 8 *membership function*.

3.4.2 Bentuk Penulisan Membership Function dan Rule Base

Program kontrol *fuzzy* yang dibuat ini, *membership function* didefinisikan dan ditulis pada bagian ROM, demikian pula aturan *fuzzy* yang dibuat. Untuk definisi *input membership function*, jumlahnya sudah pasti, yaitu empat *input* dan dua *output*. Definisi aturan *fuzzy* tidak dibatasi jumlahnya selama memori mikrokontroler masih cukup.

Setiap label pada setiap *membership function* ditulis dalam bentuk empat buah konstanta, yang terdiri atas titik 1, *slope_1*, titik 2, *slope_2*, yang membentuk fungsi trapezoidal, sehingga struktur datanya 8*4 *byte* untuk setiap input. Agar lebih jelas, dapat dilihat pada gambar 3.8.

Sebagai keadaan khusus, *slope* = 0, mengartikan garis tegak lurus, *slope_2* diasumsikan sebagai *slope* negatif. Sementara untuk alokasi memori *input* yang tidak digunakan diisi dengan FFh hingga setiap *input* memiliki struktur data 8*4 *byte*. *Membership function* untuk *input* didefinisikan mulai pada alamat 50h sampai dengan C0h.



$$Slope\_1 = \frac{FFh}{B-A}$$

$$Slope\_2 = \frac{FFh}{D-C}$$

Gambar 3.8 Bentuk penulisan *input membership function*

Output membership function didefinisikan mulai pada alamat D0h sampai dengan DFh. Bentuk *output membership function* ini berupa fungsi *singleton*, dan hanya membutuhkan satu konstanta untuk mendefinisikan setiap label. Jadi untuk setiap *output*-nya hanya memerlukan 8 *byte* konstanta. Untuk alokasi memori *output* yang tidak digunakan diisi dengan FFh hingga setiap *output* ada 8 *byte* data.

Rule fuzzy ditulis mulai pada alamat E0h sampai jumlah *rule* yang ada selama memori mikrokontroler masih cukup. Bentuk penulisannya pada bagian IF (anteseden/kondisi) ditulis dalam bentuk biner 000X XAAA dimana XX menunjukkan *input* (0 - 3) dan AAA menunjukkan label (0 - 7) dari *input* XX. Setiap aturan bisa memiliki beberapa anteseden/kondisi (biasanya 2 sampai dengan 4). Bagian IF dihubungkan dengan operator AND. Bagian THEN (konsekuen/konklusi) ditulis dalam bentuk biner 1000 YCCC dimana MSB di-*set* menunjukkan bagian konsekuen/konklusi, Y menunjukkan *output* (0 - 1) dan CCC menunjukkan label (0 - 7) dari *output* Y. Penulisan *rule* diakhiri dengan byte FFh.

Pada tugas Akhir ini *membership function* dan *rule fuzzy* didefinisikan seperti pada program *asembly* di bawah dengan aturan seperti diuraikan di atas.

```
INPUT_MFS                       ; Input membership functions
IN0MF                           ; (0) Error
      DB    000H,0FFH,060H,00BH     ;     (0) Neg_Jauh
      DB    060H,00BH,078H,020H     ;     (1) Neg_Dekat
      DB    078H,020H,081H,020H     ;     (2) Nol
      DB    081H,020H,089H,00BH     ;     (3) Pos_Dekat
      DB    089H,00BH,0FFH,0FFH     ;     (4) Pos_Jauh
      DB    0FFH,0FFH,0FFH,0FFH     ;     (5) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (6) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (7) Unused
IN1MF                           ; (1) DError
      DB    000H,0FFH,079H,02AH     ;     (0) Neg
      DB    079H,02AH,080H,02AH     ;     (1) Nol
      DB    080H,02AH,0FFH,0FFH     ;     (2) Pos
      DB    0FFH,0FFH,0FFH,0FFH     ;     (3) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (4) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (5) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (6) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (7) Unused
IN2MF                           ; (2) INPUT2
      DB    0FFH,0FFH,0FFH,0FFH     ;     (0) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (1) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (2) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (3) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (4) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (5) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (6) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (7) Unused
IN3MF                           ; (3) INPUT3
      DB    0FFH,0FFH,0FFH,0FFH     ;     (0) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (1) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (2) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (3) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (4) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (5) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (6) Unused
      DB    0FFH,0FFH,0FFH,0FFH     ;     (7) Unused

SGLTN_POS                       ; Output singleton positions

OUT0MF                                  ; (0) Daya
      DB    000H                ;     (0) Neg_Besar
      DB    040H                ;     (1) Neg_Kecil
      DB    080H                ;     (2) Nol
      DB    0CFH                ;     (3) Pos_Kecil
      DB    0FFH                ;     (4) Pos_Besar
      DB    0FFH                ;     (5) Unused
      DB    0FFH                ;     (6) Unused
      DB    0FFH                ;     (7) Unused
OUT1MF                                  ; (1) Unused
      DB    0FFH                ;     (0) Unused
      DB    0FFH                ;     (1) Unused
      DB    0FFH                ;     (2) Unused
      DB    0FFH                ;     (3) Unused
      DB    0FFH                ;     (4) Unused
      DB    0FFH                ;     (5) Unused
```

```
        DB      0FFH            ;       (6) Unused
        DB      0FFH            ;       (7) Unused

RULE_START                      ; Start of first rule

        DB      000H            ; IF Error IS Neg_Jauh
        DB      008H            ; AND DError IS Neg
        DB      084H            ; THEN Kecepatan IS Pos_Cepat

        DB      001H            ; IF Error IS Neg_Dekat
        DB      008H            ; AND DError IS Neg
        DB      084H            ; THEN Kecepatan IS Pos_Cepat

        DB      002H            ; IF Error IS Nol
        DB      008H            ; AND DError IS Neg
        DB      083H            ; THEN Kecepatan IS Pos_Lambat

        DB      003H            ; IF Error IS Pos_Dekat
        DB      008H            ; AND DError IS Neg
        DB      082H            ; THEN Kecepatan IS Nol

        DB      004H            ; IF Error IS Pos_Jauh
        DB      008H            ; AND DError IS Neg
        DB      081H            ; THEN Kecepatan IS Neg_Lambat

        DB      000H            ; IF Error IS Neg_Jauh
        DB      009H            ; AND DError IS Nol
        DB      084H            ; THEN Kecepatan IS Pos_Cepat

        DB      001H            ; IF Error IS Neg_Dekat
        DB      009H            ; AND DError IS Nol
        DB      083H            ; THEN Kecepatan IS Pos_Lambat

        DB      002H            ; IF Error IS Nol
        DB      009H            ; AND DError IS Nol
        DB      082H            ; THEN Kecepatan IS Nol

        DB      003H            ; IF Error IS Pos_Dekat
        DB      009H            ; AND DError IS Nol
        DB      081H            ; THEN Kecepatan IS Neg_Lambat

        DB      004H            ; IF Error IS Pos_Jauh
        DB      009H            ; AND DError IS Nol
        DB      080H            ; THEN Kecepatan IS Neg_Cepat

        DB      000H            ; IF Error IS Neg_Jauh
        DB      00AH            ; AND DError IS Pos
        DB      083H            ; THEN Kecepatan IS Pos_Lambat

        DB      001H            ; IF Error IS Neg_Dekat
        DB      00AH            ; AND DError IS Pos
        DB      082H            ; THEN Kecepatan IS Nol

        DB      002H            ; IF Error IS Nol
        DB      00AH            ; AND DError IS Pos
        DB      081H            ; THEN Kecepatan IS Neg_Lambat

        DB      003H            ; IF Error IS Pos_Dekat
        DB      00AH            ; AND DError IS Pos
```

```
DB      080H                ; THEN Kecepatan IS Neg_Cepat

DB      004H                ; IF Error IS Pos_Jauh
DB      00AH                ; AND DError IS Pos
DB      080H                ; THEN Kecepatan IS Neg_Cepat
```

3.4.3 Fuzzifikasi

Dari empat masukan yang masuk dan tersimpan di *register*, diproses sesuai dengan *input membership function* yang ada sehingga menghasilkan bilangan *fuzzy*. Pada setiap variabel masukan terdiri atas delapan *byte* data berupa nilai derajat keanggotaan untuk masing-masing label dalam variabel masukan tersebut. Sehingga menghasilkan data 4*8 *byte* yang disimpan pada RAM pada alamat 30h – 4Fh.

Nilai 0 menunjukkan derajat keanggotaan nol, dan nilai FFh menunjukkan derajat keanggotaan 1 (penuh). Sebagai contoh nilai 50h menunjukkan derajat keanggotaan 50h/FFh = 0,3125.

3.4.4 *Rule Evaluation*

Setelah proses fuzzifikasi selesai, maka hasilnya diproses sesuai dengan *rule* yang ada, sehingga menghasilkan keluaran. Keluaran dalam bentuk *fuzzy* berupa nilai-nilai derajat keanggotaan dari tiap-tiap label pada keluaran. Data ini berupa data 8 *byte* untuk masing-masing keluaran dan disimpan dalam RAM pada alamat 20h sampai dengan 2Fh.

3.4.5 Defuzzifikasi

Hasil pada output dalam bentuk *fuzzy* selanjutnya diubah menjadi bentuk *crisp* lagi. Metode yang digunakan adalah *Center of Grafity for Singleton* (COGS) dengan rumus

$$\text{COGS} = \frac{\sum_{i=1}^{n} Fi * Si}{\sum_{i=1}^{n} Fi}$$

dimana : F = nilai derajat keanggotaan keluaran *fuzzy*

n = jumlah *singleton* pada keluaran

S = nilai *singleton*.

Hasil dari defuzzifikasi ini merupakan keluaran kontrol *fuzzy* dan disimpan pada *register* mikrokontroler mulai pada alamat 1Ch.

3.4.6 Pembangkit PWM

Keluaran *fuzzy* pada tugas akhir ini digunakan untuk mengatur kecepatan dan arah putaran motor DC dengan cara PWM. Untuk mengoptimalkan penggunaan mikrokontroler, maka pembangkit PWM ini menggunakan pemrograman perangkat lunak mikrokontroler.

Pembangkitan ini memanfaatkan *timer* 0 pada mode 3 dengan kontrol internal. *Timer* 0 dalam mode 3 menetapkan TL0 dan TH0 sebagai dua *counter* terpisah. TL0 menggunakan kontrol bit *timer* 0 yaitu C/T, GATE, TR0, INT0, dan TF0. TH0 ditetapkan sebagai fungsi *timer*. Mode 3 sangat bermanfaat untuk aplikasi yang membutuhkan *timer/counter* ekstra 8-bit. Dengan *timer* 0 dalam mode 3, mikrokontroler seperti memiliki 3 *timer/counter*. Saat *timer* 0 dalam

mode 3, *timer* 1 dapat dihidupkan atau dimatikan, atau dapat digunakan oleh *port* serial sebagai pembangkit *baud rate.*[11]

TH0 dimanfaatkan sebagai *counter* 256, sementara TL0 difungsikan sebagai *counter* untuk keadaan *high* sesuai dengan data yang akan diubah menjadi PWM. Program untuk pembangkit PWM dibuat seperti tertera pada program di bawah ini.

```
        ORG     0BH
        MOV     TL0,#00H
        SETB    P1.0            ;Output HIGH
        RETI

        ORG     1BH
        CLR     TF0
        MOV     TL0,P2
        CLR     P1.0            ;Output LOW
        RETI

        ...
        ...
        ...

pwm:    CLR     P1.0
        MOV     IE,#8AH
        MOV     IP,#08H
        MOV     TMOD,#03H
        MOV     TH0,#00H        ;Counter 255
        MOV     TL0,#00H        ;Counter untuk keadaan HIGH
        SETB    TR1
        SETB    TR0
```

[11] Malik, Moh. Ibnu, dan Anistardi, *Bereksperimen dengan Mikrokontroler 8031* (Jakarta : P.T. Gramedia, 1997), p.23

BAB IV

PENGUJIAN DAN PENGUKURAN

Sistem kontrol gerakan tabung sinar-x pada tomograf yang dibuat terdiri atas beberapa blok sistem. Setiap blok merupakan suatu rangkaian dengan fungsi yang telah ditentukan. Pada bab ini dipaparkan mengenai hasil pengujian dan pengukuran yang telah dilakukan pada sistem, baik secara terpisah pada setiap blok maupun sebagai satu kesatuan sistem.

Untuk pengujian sistem pada setiap bloknya, maka dilakukan pemberian masukan secara manual ataupun secara simulasi. Kemudian dari keluaran yang diperoleh dapat diketahui apakah blok-blok sistem tersebut telah berfungsi sesuai dengan yang direncanakan atau belum. Apabila blok-blok tersebut telah berfungsi sesuai dengan yang diinginkan, selanjutnya blok-bok tersebut digabungkan menjadi satu sistem sesuai dengan yang direncanakan. Selanjutnya dilakukan pengujian terhadap sistem keseluruhan untuk mengetahui apakah sistem telah berjalan sesuai dengan yang direncanakan.

4.1 Pengujian Sensor Posisi

Pengukuran keluaran dari resistor variabel dilakukan dengan menggeser posisi benda dari ujung kiri sampai dengan ujung kanan. Pengukuran tegangan dilakukan pada setiap posisi sudut 15 derajat pada lintasan sepanjang 1 meter (60 derajat). Dari pengujian tersebut, sensor posisi telah dapat bekerja dan didapatkan hasil seperti pada tabel 4.1. dengan Vcc terukur 4,88 Volt.

Tabel 4.1 Data pengujian sensor posisi

Posisi (cm)	Tegangan (Volt)
-30	0,80
-15	1,60
0	2,44
15	3,28
30	4,08

4.2 Pengujian ADC

ADC diberi tegangan referensi 0 Volt untuk *Vref-* dan tegangan ujung sensor posisi 4,88 Volt untuk *Vref+*. Masukan berupa tegangan yang dihasilkan oleh sensor posisi. Pengujian dilakukan dengan memberikan masukan dari sensor posisi pada setiap pergeseran posisi 15 derajat. Dari pengujian ini diperoleh hasil seperti pada tabel 4.2.

Tabel 4.2 Data pengujian ADC

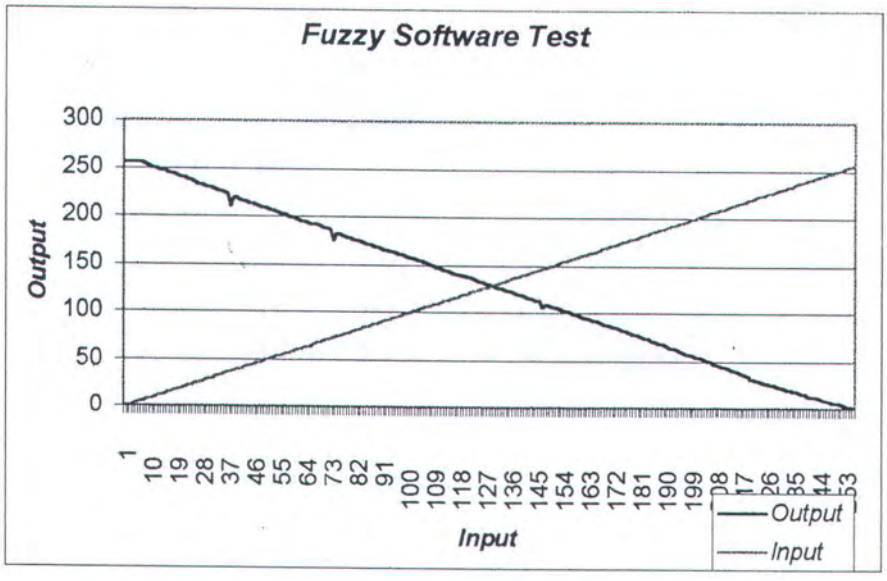
Posisi (derajat)	Keluaran ADC
-60	00101010
-30	01010100
0	10000000
30	10101100
60	11010110

4.3 Pengujian Perangkat Lunak

Pengujian perangkat lunak yang dilakukan adalah pengujian unjuk kerja proses *fuzzy*. Pengujian dilakukan menggunakan program simulasi untuk mikrokontroler keluarga MCS-51, yaitu progran AVSIM51. Untuk pengujian ini dibuat *membership function* dan aturan sederhana seperti di bawah ini :

```
IN0MF                                          ;(0) INPUT0
      DB    000H,000H,000H,007H        ;      (0) I1
      DB    000H,007H,024H,007H        ;      (1) I2
      DB    024H,007H,048H,007H        ;      (2) I3
      DB    048H,007H,06DH,007H        ;      (3) I4
      DB    06DH,007H,091H,007H        ;      (4) I5
      DB    091H,007H,0B6H,007H        ;      (5) I6
      DB    0B6H,007H,0DAH,007H        ;      (6) I7
      DB    0DAH,007H,0FFH,000H        ;      (7) I8
OUT0MF                                         ;(0) OUTPUT0
      DB    000H                ;      (0) O1
      DB    020H                ;      (1) O2
      DB    048H                ;      (2) O3
      DB    06CH                ;      (3) O4
      DB    090H                ;      (4) O5
      DB    0B4H                ;      (5) O6
      DB    0D8H                ;      (6) O7
      DB    0FFH                ;      (7) O8
RULE_START                                     ;Aturan Fuzzy

RULE_0          DB    08H,8AH      ; IF input=I0 THEN Output=O7
RULE_1          DB    09H,8BH      ; IF input=I1 THEN Output=O6
RULE_2          DB    0AH,8CH      ; IF input=I2 THEN Output=O5
RULE_3          DB    0BH,8BH      ; IF input=I3 THEN Output=O4
RULE_4          DB    0CH,8AH      ; IF input=I4 THEN Output=O3
RULE_5          DB    0DH,89H      ; IF input=I5 THEN Output=O2
RULE_6          DB    0EH,88H      ; IF input=I6 THEN Output=O1
RULE_7          DB    0FH,89H      ; IF input=I7 THEN Output=O0
```

Dari simulasi tersebut diperoleh data seperti gambar di bawah.



Gambar 4.1 Hasil pengujian program *fuzzy*

Pengujian program dengan program simulasi AVSIM51 menunjukkan kecepatan proses sistem secara keseluruhan, dengan dua *input fuzzy* dan satu *output fuzzy* serta 15 aturan *fuzzy*, membutuhkan waktu rata-rata 2,5 ms untuk satu kali proses lengkap. Pengujian dengan satu *input fuzzy* dan satu *output fuzzy* serta 5 aturan *fuzzy*, membutuhkan waktu 1383 μs sampai dengan 1483 μs. Pengujian dengan empat *input fuzzy* dan dua *output fuzzy* serta 5 aturan *fuzzy*, membutuhkan waktu 2837 μs sampai dengan 2985 μs.

4.4 Pengujian Perangkat keras

Pengujian ini dilakukan dengan cara memberikan masukan data digital yang mewakili posisi pada mikrokontroler. Bersamaan ini pula dilakukan pengujian tombol kontrol dan pembangkit PWM. Dari pengujian ini, perangkat keras telah bekarja sesuai yang direncanakan. *Set point* diisi dengan posisi tengah (80H = 10000000B), dan data posisi disimulasikan dengan *DIP switch*.

Tabel 4.3 Data pengujian *driver* motor

Masukan Posisi	Keluaran Tegangan (Volt)
00000000	8,32
00100000	7,38
01000000	6,42
01100000	5,47
10000000	0,00
10000000	-5,91
10100000	-6,79
11000000	-7,59
11100000	-8,47

4.5 Pengujian Sistem

Pengujian sistem secara keseluruhan dilakukan setelah pengujian pada setiap blok sistem selesai dan sesuai dengan yang direncanakan. Dari pengujian sistem ini diperoleh hasil pengukuran kesalahan posisi pada kondisi gerakan dari sudut-sudut posisi yang ada seperti tabel 4.4 di bawah. Selain itu dilakukan pengukuran lama waktu yang diperlukan untuk melakukan pergerakan tanpa beban dengan hasil seperti pada tabel 4.5 di bawah.

Tabel 4.4 Pengukuran kesalahan posisi

Pergerakan	Kesalahan Posisi (cm)
0^0 ke -15^0	0,4
-15^0 ke 0^0	0
0^0 ke $+15^0$	-0,8
$+15^0$ ke 0^0	-0,5
$+15^0$ ke -15^0	-0,5
-15^0 ke $+15^0$	2,3
0^0 ke -30^0	0,6
-30^0 ke 0^0	0,5
0^0 ke $+30^0$	1,5
$+30^0$ ke 0^0	1,2
$+30^0$ ke -30^0	1,5
-30^0 ke $+30^0$	3,5

Kesalahan yang terjadi disebabkan karena sensor posisi dan sumber tegangan yang kurang stabil, sehingga mempengaruhi data posisi sesaat yang masuk ke kontrol *fuzzy* melalui ADC. Sumber tegangan yang kurang stabil menyebabkan daya yang diberikan pada motor juga kurang stabil.

Tabel 4.5 Pengukuran waktu pergerakan tabung sinar-x

Pergerakan	Waktu (detik)
$+15^0$ ke -15^0	1,20
-15^0 ke $+15^0$	1,12
$+30^0$ ke -30^0	1,68
-30^0 ke $+30^0$	1,60

Dari tabel waktu pergerakan, dan panjang lintasan 47cm (untuk pergerakan 30^0), 90 cm (untuk pergerakan 60^0) dapat dihitung bahwa kecepatan rata-rata dari pergerakan tabung sinar-x pada alat yang dibuat adalah 0,41 m/s (untuk pergerakan 30^0) dan 0,55 m/s47cm (untuk pergerakan 60^0).

Untuk mengatur kecepatan pergerakan tabung sinar-x dilakukan dengan mengatur daya yang diberikan pada motor penggerak oleh keluaran kontrol *fuzzy* berupa sinyal PWM . Sinyal PWM ini digunakan sebagai masukan tegangan jangkar motor dc.

BAB V

PENUTUP

5.1 Kesimpulan

Dari perencanaan dan pembuatan kontrol gerakan tabung sinar-x ini dapat diambil kesimpulan bahwa :

1. Dengan memanfaatkan pemrograman mikrokontroler AT89C51 secara optimal, maka dapat dibuat sebuah kontroler *fuzzy* dengan kecepatan proses rata-rata 2,5 ms per siklus untuk empat masukan dan dua keluaran *fuzzy* serta lima *rule fuzzy*.
2. Mikrokontroler telah diprogram untuk pembangkit sinyal PWM pada pengontrolan kecepatan motor arus searah.
3. Pengontrolan dengan logika *fuzzy* memberikan kemudahan dalam hal perencanaan sistem kontrol, karena menggunakan cara seperti cara berfikir manusia.
4. Waktu *ekspose* yang terukur dalam peralatan yang telah dibuat adalah 1,68 detik untuk pergerakan tabung 60^0 dan 1,2 detik untuk pergerakan tabung 30^0.
5. Kesalahan pengaturan posisi rata-rata 2 % terjadi karena sensor posisi yang kurang presisi dan tegangan sumber yang tidak stabil.

5.2 Saran

Untuk mengembangkan dan meningkatkan dari apa yang telah dibuat, kami menyampaikan beberapa saran :

1. Penggunaan mikrokontroler sebagai kontrol *fuzzy* perlu ditingkatkan, karena dapat mereduksi biaya dibandingkan bila menggunakan kontrol *fuzzy* jadi.
2. Perlu mengoptimalkan pembuatan perangkat lunak untuk mendapatkan hasil yang lebih baik.
3. Untuk memperoleh hasil yang memuaskan dalam merancang sistem pengaturan berbasis logika *fuzzy*, pengalaman ahli mengenai alat yang diatur sangat dibutuhkan.
4. Pengalaman pada pengaturan *plant* sangat dibutuhkan untuk mengatasi sulitnya penentuan *membership function* yang tepat.

DAFTAR PUSTAKA

1. Jamshidi Mohammad, Nader Vadiee, dan Timothy J. Ross. 1993. *Fuzzy Logic and Control ; Software and Hardware Applications.* New Jersey : Prentice-Hall, Inc.
2. Jantzen, J. 1998. *A Robustness Study of Fuzzy Control Rules.* online paper(perf). Technical University of Denmark: Dept. of Automation, http://www.iau.dtu.dk/~jj/pubs.
3. Jantzen, J. 1998. *Design of Fuzzy Controllers.* online paper : 98-E-864 (design). Technical University of Denmark: Dept. of Automation, http://www.iau.dtu.dk/~jj/pubs.
4. Jantzen, J. 1998. *Fuzzy Supervisory Control.* online paper(proc). Technical University of Denmark: Dept. of Automation, http://www.iau.dtu.dk/~jj/pubs.
5. Jantzen, J. 1998. *The Self-Organising Fuzzy Controller.* online paper(soc). Technical University of Denmark: Dept. of Automation, http://www.iau.dtu.dk/~jj/pubs.
6. Jantzen, J. 1998. *Tuning of Fuzzy PID Controllers.* online paper(fpid). Technical University of Denmark: Dept. of Automation, http://www.iau.dtu.dk/~jj/pubs.
7. Jantzen, J. 1998. *Tutorial on Fuzzy Logic.* online paper(logic). Technical University of Denmark: Dept. of Automation, http://www.iau.dtu.dk/~jj/pubs.
8. Malik, Moh. Ibnu, dan Anistardi. 1997. *Bereksperimen dengan Mikrokontroler 8031.* Jakarta : P.T. Gramedia.
9. Ogata, Katsuhito, Edi Leksono. 1991. *Teknik Kontrol Automatik Jilid 1.* Jakarta : Erlangga.
10. Peatman, John B. 1988. *Design with Microcontrollers.* Singapore : McGraw Hill.
11. Steve Marsh, Yee Wei Huang, Jim Sibigtroth,Dave Paloian, Duberly Mazuelos, Don Weiss, Jason Spielman, John Dumas, Michael Leung, Tom Tomazin and Steve Osborn.. 1992.. *Fuzzy Logic Educational Program.*

Program Komputer. Austin, Texas : Center for Emerging Computers Technologies, Motorola Cortex Communications, Inc.

12. Tortorici, M.. 1992. *Concept in Medical Radiographic Imaging*. Philadephia : W.B. Saunders Co.
13. Wang, Li-Xin . 1997. *A Course in Fuzzy System and Control*. New Jersey : Prentice-Hall, Inc.
14. _____. 1988. *Embedded Controller Handbook*. Santa Clara : Intel.
15. _____. 1997. *International Standard IEC 1131 - Programmable Controllers, Part 7 - Fuzzy Control Programming*. Committee Draft CD 1.0 (Rel. 19 Jan 97). Frankfurt.
16. _____.1995 . *National Power ICs Databook*. Crawfordsville : National Semiconductor.

LISTING PROGRAM

```
        ORG   0000H

        AJMP  START


        ORG   03H          ; Penekanan tombol
        MOV   A,P1
        CPL   A
        MOV   0DH,A
        AJMP  INTR

        ORG   0BH
        MOV   TL0,#00H
        SETB  P0.6         ; PWM Output HIGH
        RETI

        ORG   1BH
        CLR   TF0
        MOV   TL0,0AH      ; Input PWM
        CLR   P0.6         ; PWM Output LOW
        RETI

;       Data structureables

        ORG   0015H        ; Beginning of RAM
SUM_OF_PROD  DS    3       ; 19-bit sum of products
CURRENT_INS  DS    4       ; Storage for 4 8-bit inputs
COG_OUTS     DS    2       ; Defuzzified outputs
SUM_OF_FUZ   DS    2       ; 11-bit sum of fuzzy outs
FUZ_OUTS     DS    16      ; Storage for fuzzy outputs
FUZ_INS      DS    32      ; Storage for fuzzy inputs
COGDEX       EQU   10H     ; Current out # for COG loop
LP_COUNT     EQU   11H     ; Index for fuz loop
SUMDEX       EQU   12H     ; Index for sum loop

        ORG   0030H        ; Beginning of EEPROM

INPUT_MFS                          ; Input membership functions
IN0MF                              ; (0) Error
        DB    000H,0FFH,060H,00BH  ;     (0) Neg_Jauh
        DB    060H,00BH,078H,020H  ;     (1) Neg_Dekat
        DB    078H,020H,081H,020H  ;     (2) Nol
        DB    081H,020H,089H,00BH  ;     (3) Pos_Dekat
        DB    089H,00BH,0FFH,0FFH  ;     (4) Pos_Jauh
        DB    0FFH,0FFH,0FFH,0FFH  ;     (5) Unused
        DB    0FFH,0FFH,0FFH,0FFH  ;     (6) Unused
        DB    0FFH,0FFH,0FFH,0FFH  ;     (7) Unused
IN1MF                              ; (1) DError
        DB    000H,0FFH,079H,02AH  ;     (0) Neg
        DB    079H,02AH,080H,02AH  ;     (1) Nol
        DB    080H,02AH,0FFH,0FFH  ;     (2) Pos
        DB    0FFH,0FFH,0FFH,0FFH  ;     (3) Unused
        DB    0FFH,0FFH,0FFH,0FFH  ;     (4) Unused
        DB    0FFH,0FFH,0FFH,0FFH  ;     (5) Unused
```

```
        DB    0FFH,0FFH,0FFH,0FFH     ;       (5) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (6) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (7) Unused
IN2MF                               ; (2) INPUT2
        DB    0FFH,0FFH,0FFH,0FFH     ;       (0) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (1) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (2) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (3) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (4) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (5) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (6) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (7) Unused
IN3MF                               ; (3) INPUT3
        DB    0FFH,0FFH,0FFH,0FFH     ;       (0) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (1) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (2) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (3) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (4) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (5) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (6) Unused
        DB    0FFH,0FFH,0FFH,0FFH     ;       (7) Unused

SGLTN_POS                           ; Output singleton positions
OUT0MF                              ; (0) Kecepatan
        DB    00AH                  ;       (0) Neg_Cepat
        DB    044H                  ;       (1) Neg_Lambat
        DB    080H                  ;       (2) Nol
        DB    0C0H                  ;       (3) Pos_Lambat
        DB    0FFH                  ;       (4) Pos_Cepat
        DB    0FFH                  ;       (5) Unused
        DB    0FFH                  ;       (6) Unused
        DB    0FFH                  ;       (7) Unused
OUT1MF                              ; (1) Unused
        DB    0FFH                  ;       (0) Unused
        DB    0FFH                  ;       (1) Unused
        DB    0FFH                  ;       (2) Unused
        DB    0FFH                  ;       (3) Unused
        DB    0FFH                  ;       (4) Unused
        DB    0FFH                  ;       (5) Unused
        DB    0FFH                  ;       (6) Unused
        DB    0FFH                  ;       (7) Must fill 8 per output

RULE_START                          ; Start of first rule
        DB    000H                  ; IF Error IS Neg_Jauh
        DB    008H                  ; AND DError IS Neg
        DB    084H                  ; THEN Kecepatan IS Pos_Cepat

        DB    001H                  ; IF Error IS Neg_Dekat
        DB    008H                  ; AND DError IS Neg
        DB    084H                  ; THEN Kecepatan IS Pos_Cepat

        DB    002H                  ; IF Error IS Nol
        DB    008H                  ; AND DError IS Neg
        DB    083H                  ; THEN Kecepatan IS Pos_Lambat

        DB    003H                  ; IF Error IS Pos_Dekat
        DB    008H                  ; AND DError IS Neg
        DB    082H                  ; THEN Kecepatan IS Nol
```

```
        DB    004H              ; IF Error IS Pos_Jauh
        DB    008H              ; AND DError IS Neg
        DB    081H              ; THEN Kecepatan IS Neg_Lambat

        DB    000H              ; IF Error IS Neg_Jauh
        DB    009H              ; AND DError IS Nol
        DB    084H              ; THEN Kecepatan IS Pos_Cepat

        DB    001H              ; IF Error IS Neg_Dekat
        DB    009H              ; AND DError IS Nol
        DB    083H              ; THEN Kecepatan IS Pos_Lambat

        DB    002H              ; IF Error IS Nol
        DB    009H              ; AND DError IS Nol
        DB    082H              ; THEN Kecepatan IS Nol

        DB    003H              ; IF Error IS Pos_Dekat
        DB    009H              ; AND DError IS Nol
        DB    081H              ; THEN Kecepatan IS Neg_Lambat

        DB    004H              ; IF Error IS Pos_Jauh
        DB    009H              ; AND DError IS Nol
        DB    080H              ; THEN Kecepatan IS Neg_Cepat

        DB    000H              ; IF Error IS Neg_Jauh
        DB    00AH              ; AND DError IS Pos
        DB    083H              ; THEN Kecepatan IS Pos_Lambat

        DB    001H              ; IF Error IS Neg_Dekat
        DB    00AH              ; AND DError IS Pos
        DB    082H              ; THEN Kecepatan IS Nol

        DB    002H              ; IF Error IS Nol
        DB    00AH              ; AND DError IS Pos
        DB    081H              ; THEN Kecepatan IS Neg_Lambat

        DB    003H              ; IF Error IS Pos_Dekat
        DB    00AH              ; AND DError IS Pos
        DB    080H              ; THEN Kecepatan IS Neg_Cepat

        DB    004H              ; IF Error IS Pos_Jauh
        DB    00AH              ; AND DError IS Pos
        DB    080H              ; THEN Kecepatan IS Neg_Cepat

END_OF_RULE  DB    0FFH

;       Program kontrol pergerakan motor

        ORG   100H
INTR:
SD60:   MOV   A,#00001000B      ; Tekan 60  (S1)
        ANL   A,0DH
        JZ    SD30
        ORL   A,0EH
        ANL   A,#00001111B
        MOV   0EH,A
        SJMP  INTRA
SD30:   MOV   A,#00010000B      ; Tekan 30  (S2)
        ANL   A,0DH
```

```
        JZ    KANAN
        ORL   A,0EH
        ANL   A,#00010111B
        MOV   0EH,A
        SJMP  INTRA
KANAN:  MOV   A,#00000001B      ; Tekan --> (S3)
        ANL   A,0DH
        JZ    TENGAH
        ORL   A,0EH
        ANL   A,#00011001B
        MOV   0EH,A
        ACALL SETPOINT
        SJMP  INTRA
TENGAH: MOV   A,#00000010B      ; Tekan |    (S4)
        ANL   A,0DH
        JZ    KIRI
        ORL   A,0EH
        ANL   A,#00011010B
        MOV   0EH,A
        ACALL SETPOINT
        SJMP  INTRA
KIRI:   MOV   A,#00000100B      ; Tekan <-- (S5)
        ANL   A,0DH
        JZ    EXPOSE
        ORL   A,0EH
        ANL   A,#00011100B
        MOV   0EH,A
        ACALL SETPOINT
        SJMP  INTRA
EXPOSE: MOV   A,#00100000B      ; Tekan Ekspose   (S0)
        ANL   A,0DH
        JZ    INTRA
        MOV   A,0EH
        ANL   A,#00000010B
        JZ    EXP1
        SJMP  INTRA
EXP1:   MOV   A,0EH
        XRL   A,#00000101B
        MOV   0EH,A
        ACALL SETPOINT
INTRA:
        MOV   P0,0EH
        RETI

SETPOINT:                       ; Input Set Point
KR60:   CJNE A,#00001100B,KR30
        MOV   0FH,#00010110B    ; Posisi -60 derajat
        SJMP  SPA
KR30:   CJNE A,#00010100B,KN30
        MOV   0FH,#01001000B    ; Posisi -30 derajat
        SJMP  SPA
KN30:   CJNE A,#00010001B,KN60
        MOV   0FH,#10110010B    ; Posisi +30 derajat
        SJMP  SPA
KN60:   CJNE A,#00001001B,TG
        MOV   0FH,#11100100B    ; Posisi +60 derajat
        SJMP  SPA
TG:     MOV   0FH,#01111110B    ; Posisi Tengah
SPA:    RET
```

```
;       Program utama, inisialisasi, pembangkit PWM, dan I/O

START:  MOV   SP,#70H           ; Stack Pointer mulai alamat 70H
        CLR   P0.6
        CLR   P0.7
        MOV   IE,#8FH           ; Interrupt Enable kecuali serial
        MOV   IP,#0AH           ; Prioritas pada T0 dan T1
        SETB  TCON.0
        MOV   TMOD,#03H
        MOV   TH0,#00H          ; Counter 255
        MOV   TL0,#00H
        SETB  TR1
        SETB  TR0

INIT:   SETB  P3.0              ; /RD & /CS ADC = 1
        MOV   1CH,#80H          ; Output 80H
        MOV   0FH,#7EH          ; Posisi Tengah
        MOV   0EH,#0AH          ; Posisi Tengah & Sudut 60
        MOV   P0,0EH            ; Display kontrol
        CLR   P3.0              ; /RD & /CS ADC = 0
;       MOV   R6,#16            ; Counter untuk DError
;       MOV   19H,#80H          ; DError = 0

OUTPUT: SETB  P3.0              ; /RD & /CS ADC = 1
        MOV   08H,TH1           ; ****************************
        MOV   09H,TL1           ; *   Counter seluruh program *
        MOV   TL1,#00H          ; *                           *
        MOV   TH1,#00H          ; ****************************
        MOV   0BH,1CH           ; Normalisasi Output sbg Input PWM
        MOV   A,#80H
        CLR   P3.0              ; /RD & /CS ADC = 0 --> Read ADC
        CJNE  A,0BH,NORM        ; Titik nol = 80H
        CLR   IE.3              ; Non aktifkan timer 1
        SETB  P0.6
        SJMP  NORM2
NORM:   SETB  IE.3              ; Aktifkan timer 1
        JNC   NORM1             ; Jika > 80H berarti arah kanan
        MOV   A,0BH             ; Output > 80H
        CLR   C
        SUBB  A,#80H
        MOV   B,#2
        MUL   AB
        MOV   0AH,A             ; Input PWM disimpan di 0AH
        CLR   P0.7              ; Arah geser kanan
        SJMP  INPUT
NORM1:  CLR   C                 ; Output < 80H
        MOV   A,#2
        MOV   B,0BH
        MUL   AB
        MOV   0AH,A             ; Input PWM disimpan di 0AH
NORM2:  SETB  P0.7              ; Arah geser kiri

INPUT:  MOV   A,0FH             ; A = Set Point
        MOV   0CH,P2            ; Hasil ADC simpan di alamat 0CH
        CJNE  A,0CH,KURANG1
KURANG1:JC    LEBIH1            ; A < data ? Ya
        SUBB  A,0CH             ; No
        MOV   B,#2
```

```
        DIV   AB                ; Error = 128 - A/2
        MOV   0CH,#80H
        XCH   A,0CH
        SUBB  A,0CH
        SJMP  TERUS
LEBIH1: XCH   A,0CH             ; A <= data
        CLR   C
        SUBB  A,0CH
        MOV   B,#2
        DIV   AB                ; Error = 128 + A/2
        ADD   A,#80H
TERUS:
;       CJNE  R6,#16,DECNT0     ; Input1 dihitung sekali setiap 16
loop
        MOV   R7,18H            ; R7 = Error_Lama
;DECNT0:     DEC   R6
        MOV   18H,A             ; Isi Input0 dengan Error Posisi
;       CJNE  R6,#0,FUZZIFY     ; Apa sudah 16 loop ?
;       MOV   R6,#16            ; Isi counter input1 dengan 16
        XCH   A,R7              ;
        CJNE  A,07H,KURANG2     ; A = Error ; 07H = R7
KURANG2:JC    LEBIH2            ; A < R7 ? Ya
        SUBB  A,R7              ; No
        MOV   B,#2
        MUL   AB                ; DError = 128 - 2*(Error-
Error_Lama)
        MOV   R7,#80H
        XCH   A,R7
        SUBB  A,R7
        SJMP  TERUS1
LEBIH2: XCH   A,R7              ; A <= data
        CLR   C
        SUBB  A,R7
        MOV   B,#2
        MUL   AB                ; DError = 128 + 2*(Error-
Error_Lama)
        ADD   A,#80H
TERUS1: MOV   19H,A             ; Isi Input1 dengan DError

;       Fuzzy Inference Engine Starts Here

FUZZIFY:

        MOV   DPTR,#FUZ_INS     ; Point at fuzzy input table
        MOV   R1,DPL
                                ; Point at current input values
;       PUSH  1BH               ; Put input values on stack
;       PUSH  1AH               ; Digunakan hanya 1 input
        PUSH  19H
        PUSH  18H
        MOV   DPTR,#INPUT_MFS   ; Point at input MF specs
NXTIN_LP:
        POP   02H               ; Get next current input value
        MOV   R0,02H            ; For 8 loop passes
        MOV   R3,#08H           ; Initialize loop counter
GRAD_LOOP:                      ; Get grade for 1 laof 1 input
GET_GRADE:
        MOV   R0,02H
        MOV   B,#00H            ; In case grade = 0
```

```
        MOV   A,#2
        MOVC  A,@A+DPTR
        CJNE  A,00H,LOOP1       ; Input value - pt2 -> A
        SJMP  NOT_SEG2
LOOP1:
        JNC   NOT_SEG2          ; If input < pt2
        CLR   C
        XCH   A,R0
        SUBB  A,R0
        MOV   B,A
        MOV   A,#3
        MOVC  A,@A+DPTR         ; Slope 2
        JZ    VERT_SLP          ; Skip if vert slope
        MUL   AB                ; (In - pt2) * slp2 -> A:B
        XCH   A,B
        JZ    NO_FIX                ; Check for > $FF
                                ; If upper 8 = 0
        MOV   B,#00H                ; Limit grade to 0
        SJMP  HAV_GRAD          ; In limit region of seg 2
NO_FIX:
        MOV   A,#0FFH               ; B - $FF
        SUBB  A,B               ; $FF - B
        MOV   B,A
        SJMP  HAV_GRAD          ; ($FF -((In - pt2) * slp2))
NOT_SEG2:
        MOV   R0,02H
        CLR   A
        MOVC  A,@A+DPTR
        CJNE  A,00H,LOOP2       ; Input value - pt1
        SJMP  LOOP3             ; Input value = pt1
LOOP2:
        JNC   HAV_GRAD          ; In < pt1 so grade = 0
LOOP3:
        CLR   C
        XCH   A,R0
        SUBB  A,R0
        MOV   B,A               ; Input value > pt1
        MOV   A,#1
        MOVC  A,@A+DPTR
        JZ    VERT_SLP          ; Skip if vert slope
        MUL   AB                ; (In - pt1) * slp1 -> A:B
        XCH   A,B
        JZ    HAV_GRAD          ; Check for > $FF
                                ; Result OK in B
VERT_SLP:
        MOV   B,#0FFH           ; Limit region or vert slope
HAV_GRAD:
        INC   DPTR              ; Point at next MF spec
        INC   DPTR
        INC   DPTR
        INC   DPTR
        MOV   @R1,B             ; Save @ fuzzy input
        INC   R1                ; Point at next
        DJNZ  R3,GRAD_LOOP      ; For 8 labels of 1 input
        CJNE  R1,#40H,LOOP4     ; Selesai 2(#40H) fuzzy ins ;
(4=#50H)
        SJMP  LOOP5             ; Done with all fuzzy ins?
LOOP4:
        AJMP  NXTIN_LP          ; If not, process next input
```

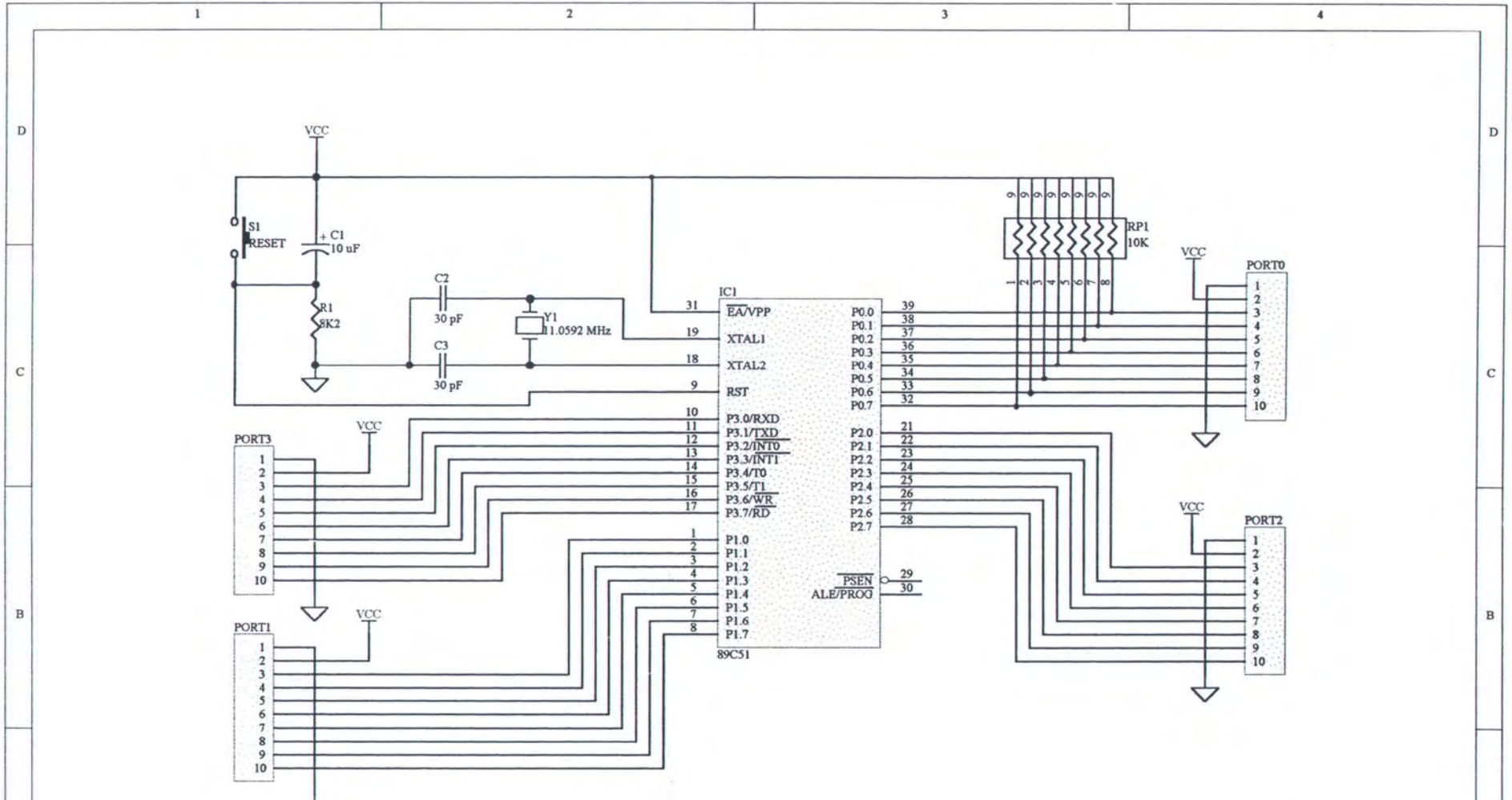
```
;       Done with fuzzification, evaluate rules next

LOOP5:
        MOV   DPTR,#FUZ_OUTS    ; Point at first fuzzy output
        MOV   R0,DPL
        MOV   A,#16             ; 16 fuzzy outputs
CLR_OUTS:
        MOV   @R0,#00H          ; Clear a fuzzy output
        INC   R0                ; Point at next
        DEC   A                 ; Loop index
        JNZ   CLR_OUTS          ; Continue till all fuzzy outs 0
        MOV   DPTR,#RULE_START  ; Point to start of 1st rule
RULE_TOP:
        MOV   R2,#0FFH          ; Begin processing a rule string

;       A will hold grade of smallest (min) if part

IF_LOOP:
        CLR   A
        MOVC  A,@A+DPTR         ; Get rule byte 000X XAAA; If X is A
        RLC   A
        JC    THEN_LOOP         ; If MSB=1, exit to then loop
        RRC   A
        INC   DPTR              ; Point at next rule byte
        MOV   R0,#30H                 ; Point at fuzzy inputs
        ADD   A,R0              ; Point to specific fuzzy input
        MOV   R0,A
        MOV   A,@R0
        CJNE  A,02H,LOOP6       ; Is this fuzzy input lower?
        SJMP  IF_LOOP           ; If not don't replace lowest
LOOP6:
        JNC   IF_LOOP
        MOV   02H,@R0           ; Replace lowest if
        JNZ   IF_LOOP           ; Unless zero, do next rule byte
????

;A zero value fuzzy input makes rule completely not true so skip
;rest of if parts and all then parts to start of next rule

FIND_THEN:
        CLR   A
        MOVC  A,@A+DPTR         ; Get next rule byte
        RLC   A
        JC    FIND_IF           ; MSB set means its a then part
        INC   DPTR              ; Point at next rule byte
        SJMP  FIND_THEN         ; Loop till pointing at a then part
FIND_IF:
        INC   DPTR              ; Adv rule pointer to if part
        CLR   A
        MOVC  A,@A+DPTR         ; Get next rule byte
        RLC   A
        JNC   RULE_TOP          ; MSB clear means its an if part
        RRC   A
        CJNE  A,#0FFH,FIND_IF   ; $FF is no more rules marker
                                ; Continue looking for if or $FF
        SJMP  DEFUZ             ; When all rules done, go defuzzify
THEN_LOOP:
        RRC   A                 ; Restore rule byte
```

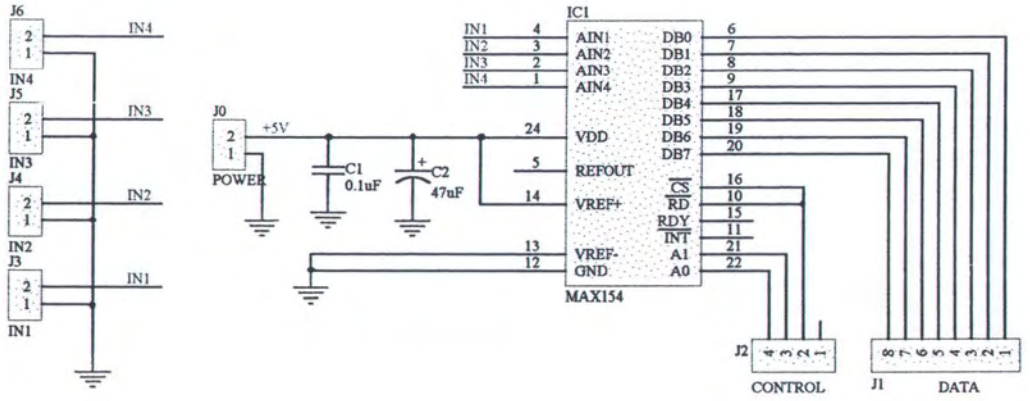
```
        MOV   R0,#20H           ; Point at fuzzy outputs
        ANL   A,#0FH            ; Save 8 times out # + label #
        ADD   A,R0
        MOV   R0,A              ; X points at fuzzy output

;       Grade of membership for rule is in A accumulator

        MOV   A,@R0
        CJNE  A,02H,LOOP7       ; Compare to fuzzy output
        SJMP  NOT_HIER          ; Branch if not higher
LOOP7:
        JNC   NOT_HIER
        MOV   @R0,02H           ; Grade is higher so update
NOT_HIER:
        INC   DPTR              ; Adv rule pointer to next byte
        CLR   A
        MOVC  A,@A+DPTR         ; Get rule byte
        RLC   A
        JNC   RULE_TOP          ; If MSB=0 its a new rule
        RRC   A
CHK_END:                        ; Check for end of rules flag
        CJNE  A,#0FFH,THEN_LOOP; If not $FF, must be a then part

;       All rules evaluated. Now defuzzify fuzzy outputs

DEFUZ:
        MOV   DPTR,#SGLTN_POS   ; Point at 1st output singleton
        MOV   03H,DPL
        MOV   04H,DPH
        MOV   DPTR,#FUZ_OUTS    ; Point at 1st fuzzy output
        MOV   R0,DPL
        MOV   COGDEX,#00H       ; Loop index will run from 0->1
COG_LOOP:
        MOV   02H,R0
        MOV   SUMDEX,#08H       ; 8 fuzzy outs per COG output
                                ; Inner loop runs 8->0
        MOV   DPTR,#SUM_OF_FUZ  ; Used for quicker clears
        MOV   R0,DPL
        MOV   @R0,#00H
        INC   R0
        MOV   @R0,#00H          ; Sum of fuzzy outputs
        MOV   DPTR,#SUM_OF_PROD ; Upper 8-bits of sum of products
        MOV   R0,DPL
        MOV   @R0,#00H
        INC   R0
        MOV   @R0,#00H
        INC   R0
        MOV   @R0,#00H
SUM_LOOP:
        MOV   R0,02H            ; Get a fuzzy output
        MOV   A,@R0             ; Clear upper 8-bits
        MOV   DPTR,#SUM_OF_FUZ  ; Add to sum of fuzzy outputs
        MOV   R0,DPL
        MOV   R1,DPL
        INC   R1
        ADD   A,@R1
        MOV   @R1,A
        RLC   A
        ANL   A,#01H
```

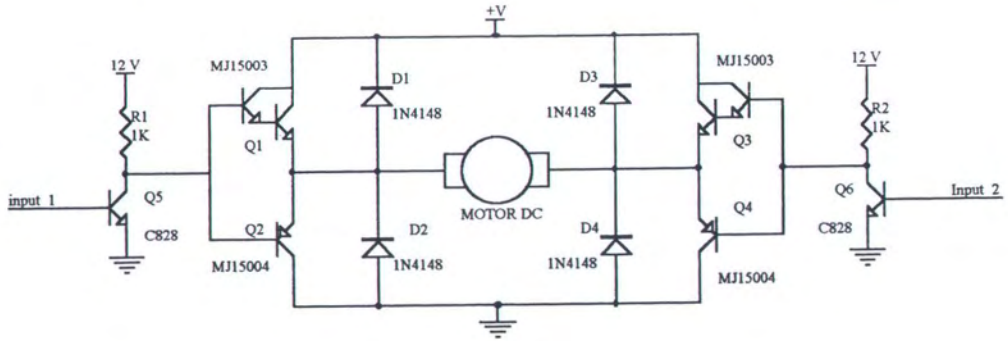
```
        ADD   A,@R0              ; Update RAM variable
        MOV   @R0,A
        MOV   R0,02H
        MOV   DPL,03H
        MOV   DPH,04H
        CLR   A
        MOVC  A,@A+DPTR          ; Get fuzzy output again
        MOV   B,@R0              ; Get Output singleton position
        MUL   AB                 ; Position times weight
        MOV   DPTR,#SUM_OF_PROD
        MOV   R0,DPL
        INC   R0                 ; Low 16-bits of sum of products
        INC   R0                 ;
        XCH   A,@R0              ; Update low 16-bits
        ADD   A,@R0
        XCH   A,@R0
        DEC   R0
        CLR   A
        RLC   A
        ADD   A,B
        ADD   A,@R0              ;
        MOV   @R0,A              ; Upper 8-bits of 24-bit sum
        DEC   R0
        CLR   A                  ;
        RLC   A                  ;Add carry from 16-bit add
        ADD   A,@R0              ;
        MOV   @R0,A              ;
        MOV   R0,02H             ;
        MOV   DPL,03H            ;
        MOV   DPH,04H            ; Point at next singleton pos.
        INC   DPTR               ;
        MOV   03H,DPL            ;
        MOV   04H,DPH            ; Point at next fuzzy output
        INC   R0                 ;
        MOV   02H,R0             ; Inner loop index
        DJNZ  SUMDEX,SUM_LOOP    ; For all labels this output

        MOV   02H,R0             ; Save index for now
        CLR   A                  ; In case divide by zero
        MOV   A,1EH              ; A=(SUM_OF_FUZ or SUM_OF_FUZ+1)
        *
        ORL   A,1FH              ; Demominator for divide
        *
        JNZ   GESER              ; Output = 0
        MOV   14H,#00H           ; Branch if denominator is 0
        AJMP  SAV_OUT

                                 ; Pembagian 19-bit dg 11-bit
GESER:                           ; Jadikan 16-bit dg 8 bit
        MOV   A,1EH
        JZ    BAGI
        MOV   R0,#1FH
        MOV   A,@R0
        SWAP  A
        MOV   @R0,A
        MOV   A,1EH
        SWAP  A
        XCHD  A,@R0
        MOV   1EH,A
        MOV   A,1FH
```

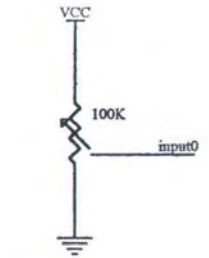
```
        RLC   A
        JNC   LP1
        INC   1EH
LP1:    MOV   @R0,1EH
        MOV   1EH,#00H
        MOV   R0,#16H
        MOV   A,@R0
        SWAP  A
        MOV   @R0,A
        MOV   A,15H
        SWAP  A
        XCHD  A,@R0
        MOV   15H,A
        MOV   R0,#17H
        MOV   A,@R0
        SWAP  A
        MOV   @R0,A
        MOV   A,16H
        XCHD  A,@R0
        MOV   16H,A
        MOV   A,@R0
        RLC   A
        JNC   LP2
        INC   16H
LP2:    MOV   @R0,16H
        MOV   16H,15H
        MOV   15H,#00H

BAGI:   MOV   14H,#00H          ; Proses Pembagian
        CLR   C
L0:     MOV   A,16H
        ADD   A,14H
        MOV   14H,A
        MOV   A,1FH
        CPL   A
        INC   A
        MOV   B,16H
        MUL   AB
        MOV   16H,B
        ADD   A,17H
        JNC   L1
        INC   16H
L1:     MOV   17H,A
        MOV   A,16H
        CJNE  A,#1H,L2
        SJMP  L0
L2:     JNC   L0
        MOV   A,17H
        MOV   B,1FH
        DIV   AB
        MOV   16H,A
        MOV   17H,B
        MOV   A,1FH
        DIV   AB
        CJNE  A,#2H,L4
        SJMP  L5
L4:     JNC   L5
        INC   16H
L5:     MOV   A,16H
```

```
        ADD   A,14H
        JC    L6
        MOV   14H,A
        SJMP  SAV_OUT
L6:     MOV   14H,#FFH

SAV_OUT:
        MOV   DPTR,#COG_OUTS    ; Point to 1st defuz output
        MOV   R0,DPL
        MOV   A,COGDEX          ; Curent output number
        ADD   A,R0
        MOV   R0,A              ; Point to correct output
        MOV   @R0,14H           ; Update defuzzified output
                                ; Curent output number
        MOV   R0,02H            ; Recover index
        INC   COGDEX            ; Increment loop index
        MOV   A,#1              ; Selesai 1(#1) outs ; (2=#2)
        CJNE  A,COGDEX,LOOP9    ; Done with two outs?
        SJMP  AKHIR
LOOP9:
        JC    AKHIR             ; If not, continue loop
        AJMP  COG_LOOP

;       Inference engine has completed one pass of all rules.

AKHIR:
        JMP   OUTPUT            ; ULANGI PROSES TOTAL
```

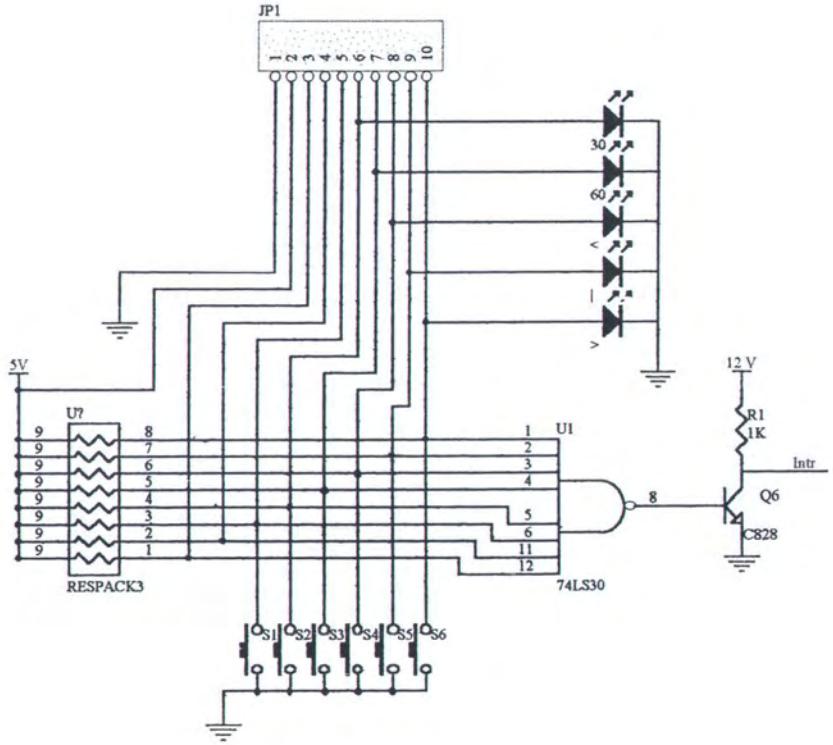

1
2
3
4
D
C
B
VCC
S1
RESET
+ C1
10 uF
R1
8K2
C2
30 pF
C3
30 pF
Y1
11.0592 MHz
IC1
EA/VPP
XTAL1
XTAL2
RST
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
89C51
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P2.0
P2.1
P2.2
P2.3
P2.4
P2.5
P2.6
P2.7
PSEN
ALE/PROG
RP1
10K
PORT0
PORT1
PORT2
PORT3


J6
IN4
IN4
J5
IN3
IN3
J4
IN2
IN2
J3
IN1
IN1
J0
+5V
POWER
C1
0.1uF
C2
47uF
IC1
IN1 4
IN2 3
IN3 2
IN4 1
AIN1
AIN2
AIN3
AIN4
VDD 24
REFOUT 5
VREF+ 14
VREF- 13
GND 12
DB0 6
DB1 7
DB2 8
DB3 9
DB4 17
DB5 18
DB6 19
DB7 20
CS 16
RD 10
RDY 15
INT 11
A1 21
A0 22
MAX154
J2
CONTROL
J1
DATA


+V
12 V
R1
1K
input 1
Q5
C828
MJ15003
Q1
Q2
MJ15004
D1
1N4148
D2
1N4148
MOTOR DC
D3
1N4148
D4
1N4148
MJ15003
Q3
Q4
MJ15004
12 V
R2
1K
Q6
C828
Input 2



Sensor Posisi



DAFTAR RIWAYAT HIDUP

MOH. SAMSUN AFIFUDDIN, dilahirkan di Kediri pada tanggal 19 Agustus 1975 merupakan putra pertama dari enam bersaudara dari Bapak Asmu'i dan Ibu Siti Maimonah, yang tinggal di Kediri Jawa Timur. Terdaftar sebagai mahasiswa Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember melalui jalur UMPTN tahun 1994, dengan NRP 2294 100 084.

Pendidikan yang telah ditempuh selama ini adalah :

- MII Banjarmlati Kediri tahun 1982 – 1988
- SMPN 1 Kediri tahun 1988-1991
- SMAN 2 Kediri tahun 1991-1994
- Institut Teknologi Sepuluh Nopember , Fakultas Teknologi Industri, Jurusan Teknik Elektro, Bidang Studi Elektronika sejak tahun 1994 yang diharapkan dapat menyelesaikan studinya Maret 2000

Selama mahasiswa penyusun aktif organisasi kemahasiswaan di JMMI - ITS. Di Bidang studi Elektronika aktif sebagai asisten praktikum Rangkaian Listrik, Praktikum Elektronika, Praktikum Elektronika Lanjut II, dan sebagai *supervisor* LAN.